DOTS   T. Reddy
Internet-Draft   McAfee
Intended status: Standards Track   M. Boucadair
Expires: ~~June 21,~~ **July 13,** 2018   Orange
                                                              P. Patil
                                                                 Cisco
                                                          A. Mortensen
                                                  Arbor Networks, Inc.
                                                             N. Teague
                                                        Verisign, Inc.
                                                    ~~December 18, 2017~~
                                                      January 9, 2018

           Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal
                                    Channel
                        ~~draft-ietf-dots-signal-channel-14~~
                        draft-ietf-dots-signal-channel-15

Abstract

   This document specifies the DOTS signal channel, a protocol for
   signaling the need for protection against Distributed Denial-of-
   Service (DDoS) attacks to a server capable of enabling network
   traffic mitigation on behalf of the requesting client.

   A companion document defines the DOTS data channel, a separate
   reliable communication layer for DOTS management and configuration
   purposes.

Editorial Note (To be removed by RFC Editor)

   Please update these statements with the RFC number to be assigned to
   this document:

   o "This version of this YANG module is part of RFC XXXX;"

   o "RFC XXXX: Distributed Denial-of-Service Open Threat Signaling
      (DOTS) Signal Channel";

   o "| 3.00 | Alternate server | [RFCXXXX] |"

   o **"| 4.16 | CUID Collision | [RFCXXXX] |"**

   o **"| 5.06 | Hop Limit Reached | [RFCXXXX] |"**

   o reference: RFC XXXX

   o This RFC
   Please update TBD statements with the port number to be assigned to
   DOTS Signal Channel Protocol.

Status of This Memo

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

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

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

   This Internet-Draft will expire on ~~June 21,~~ **July 13,** 2018.

Table of Contents

1. Introduction

   A distributed denial-of-service (DDoS) attack is an attempt to make
   machines or network resources unavailable to their intended users.
   In most cases, sufficient scale can be achieved by compromising
   enough end-hosts and using those infected hosts to perpetrate and
   amplify the attack. The victim in this attack can be an application
   server, a host, a router, a firewall, or an entire network.

   Network applications have finite resources like CPU cycles, the
   number of processes or threads they can create and use, the maximum
   number of simultaneous connections it can handle, the limited
   resources of the control plane, etc. When processing network
   traffic, such applications are supposed to use these resources to
   offer the intended task in the most efficient manner. However, a
   DDoS attacker may be able to prevent an application from performing
   its intended task by making the application exhaust its finite
   resources.

   TCP DDoS SYN-flood, for example, is a memory-exhausting attack while
   ACK-flood is a CPU-exhausting attack [RFC4987]. Attacks on the link
   are carried out by sending enough traffic so that the link becomes
   congested, thereby likely causing packet loss for legitimate traffic.
   Stateful firewalls can also be attacked by sending traffic that
   causes the firewall to maintain an excessive number of states that
   may jeopardize the firewall's operation overall, besides like
   performance impacts. The firewall then runs out of memory, and can
   no longer instantiate the states required to process legitimate
   flows. Other possible DDoS attacks are discussed in [RFC4732].

   In many cases, it may not be possible for network administrators to
   determine the cause(s) of an attack. They may instead just realize
   that certain resources seem to be under attack. This document
   defines a lightweight protocol that allows a DOTS client to request
   mitigation from one or more DOTS servers for protection against
   detected, suspected, or anticipated attacks. This protocol enables
   cooperation between DOTS agents to permit a highly-automated network
   defense that is robust, reliable, and secure.

   An example of a network diagram that illustrates a deployment of DOTS
   agents is shown in Figure 1. In this example, a DOTS server is
   operating on the access network. A DOTS client is located on the LAN
   (Local Area Network), while a DOTS gateway is embedded in the CPE
   (Customer Premises Equipment).

      Network
      Resource   CPE router   Access network   _____
    +-----------+   +-------------+   +-------------+   /   \
    | |___| | | |____| Internet |
    |DOTS client| | DOTS gateway| | DOTS server | | |
    | | | | | | | |
    +-----------+   +-------------+   +-------------+   _____/
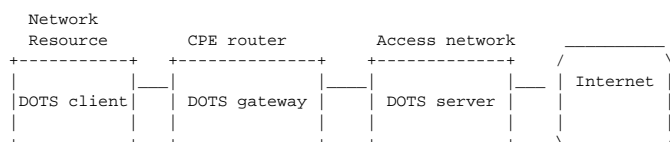
                   Figure 1: Sample DOTS Deployment (1)

   DOTS servers can also be reachable over the Internet, as depicted in
   Figure 2.

      Network   DDoS mitigation
      Resource   CPE router   _____   service
    +-----------+   +-------------+   /   \ +-------------+
    | |___| |___| |___| |
    |DOTS client| |DOTS gateway | | Internet | | DOTS server |
    | | | | | | | |
    +-----------+   +-------------+   _____/   +-------------+

                   Figure 2: Sample DOTS Deployment (2)

In typical deployments, the DOTS client belongs to a different
administrative domain than the DOTS server. For example, the DOTS
client is embedded in a firewall protecting services owned and
operated by a domain, while the DOTS server is owned and operated by
a different domain providing DDoS mitigation services. The latter
might or might not provide connectivity services to the network
hosting the DOTS client.

The DOTS server may (not) be co-located with the DOTS mitigator. In
typical deployments, the DOTS server belongs to the same
administrative domain as the mitigator. The DOTS client can
communicate directly with a DOTS server or indirectly via a DOTS
gateway.

The document adheres to the DOTS architecture
[I-D.ietf-dots-architecture]. The requirements for DOTS signal
channel protocol are documented in [I-D.ietf-dots-requirements].
This document satisfies all the use cases discussed in
[I-D.ietf-dots-use-cases].

This document focuses on the DOTS signal channel. This is a
companion document of the DOTS data channel specification
[I-D.ietf-dots-data-channel] that defines a configuration and a bulk
data exchange mechanism supporting the DOTS signal channel.

2. Notational Conventions and Terminology

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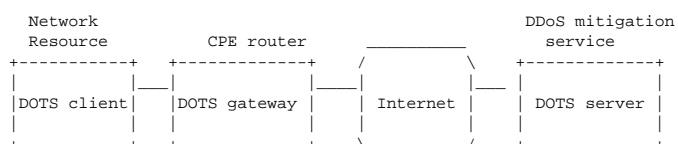"OPTIONAL" in this document are to be interpreted as described in
[RFC2119].

(D)TLS is used for statements that apply to both Transport Layer
Security [RFC5246] and Datagram Transport Layer Security [RFC6347].
Specific terms are used for any statement that applies to either
protocol alone.

The reader should be familiar with the terms defined in
[I-D.ietf-dots-architecture].

The meaning of the symbols in YANG tree diagrams is defined in
[I-D.ietf-netmod-yang-tree-diagrams].

3. Design Overview

The DOTS signal channel is built on top of the Constrained
Application Protocol (CoAP) [RFC7252], a lightweight protocol
originally designed for constrained devices and networks. The many
features of CoAP (expectation of packet loss, support for
asynchronous non-confirmable messaging, congestion control, small
message overhead limiting the need for fragmentation, use of minimal
resources, and support for (D)TLS) makes it a good candidate to build
the DOTS signaling mechanism from.

The DOTS signal channel is layered on existing standards (Figure 3).

```
            +--------------------+
            | DOTS Signal Channel |
            +--------------------+
            |         CoAP        |
            +----------+----------+
            |   TLS    |   DTLS   |
            +----------+----------+
            |   TCP    |   UDP    |
            +----------+----------+
            |         IP          |
            +--------------------+
```
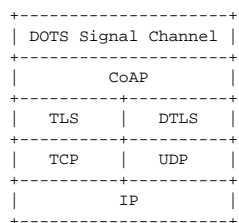
         Figure 3: Abstract Layering of DOTS signal channel over CoAP over
                               (D)TLS

By default, a DOTS signal channel MUST run over port number TBD as
defined in Section 9.1, for both UDP and TCP, unless the DOTS server
has a mutual agreement with its DOTS clients to use a different port
number. DOTS clients may alternatively support means to dynamically
discover the ports used by their DOTS servers. In order to use a
distinct port number (as opposed to TBD), DOTS clients and servers
should support a configurable parameter to supply the port number to
use. The rationale for not using the default port number 5684
((D)TLS CoAP) is to allow for differentiated behaviors in
environments where both a DOTS gateway and an IoT gateway (e.g.,
Figure 3 of [RFC7452]) are present.

The signal channel is initiated by the DOTS client (Section 4.4).
Once the signal channel is established, the DOTS agents periodically
send heartbeats to keep the channel active (Section 4.7). At any
time, the DOTS client may send a mitigation request message to a DOTS
server over the active channel. While mitigation is active because
of the higher likelihood of packet loss during a DDoS attack, the
DOTS server periodically sends status messages to the client,
including basic mitigation feedback details. Mitigation remains
active until the DOTS client explicitly terminates mitigation, or the
mitigation lifetime expires.

DOTS signaling can happen with DTLS [RFC6347] over UDP and TLS
[RFC5246] over TCP. Likewise, DOTS requests may be sent using IPv4
or IPv6 transfer capabilities. A Happy Eyeballs procedure for DOTS
signal channel is specified in Section 4.3.

Messages exchanged between DOTS agents are serialized using Concise
Binary Object Representation (CBOR) [RFC7049], CBOR is a binary
encoding scheme designed for small code and message size. CBOR-
encoded payloads are used to carry signal channel-specific payload
messages which convey request parameters and response information
such as errors. In order to allow the use of the same data models,
[RFC7951] specifies the JSON encoding of YANG-modeled data. A
similar effort for CBOR is defined in [I-D.ietf-core-yang-cbor]. All
parameters in the payload of the DOTS signal channel are mapped to
CBOR types as specified in Section 6.

From that standpoint, this document specifies a YANG data model for
representing mitigation scopes and DOTS signal channel session
configuration data (Section 5). Representing these data as CBOR data
is assumed to follow the rules in [I-D.ietf-core-yang-cbor] or those
in [RFC7951] combined with JSON/CBOR conversion rules in [RFC7049]. ⸗

In order to prevent fragmentation, DOTS agents must follow the
recommendations documented in Section 4.6 of [RFC7252]. Refer to
Section 7.3 for more details.

DOTS agents MUST support GET, PUT, and DELETE CoAP methods. The
payload included in CoAP responses with 2.xx and 3.xx Response Codes
MUST be of content type "application/cbor" (Section 5.5.1 of
[RFC7252]). CoAP responses with 4.xx and 5.xx error Response Codes
MUST include a diagnostic payload (Section 5.5.2 of [RFC7252]). The
Diagnostic Payload may contain additional information to aid
troubleshooting.

In deployments where multiple DOTS clients are enabled in a network
(owned and operated by the same entity), the DOTS server may detect
conflicting mitigation requests from these clients. This document
does not aim to specify a comprehensive list of conditions under
which a DOTS server will characterize two mitigation requests from
distinct DOTS clients as conflicting, nor recommend a DOTS server
behavior for processing conflicting mitigation requests. Those
considerations are implementation- and deployment-specific.
Nevertheless, the document specifies the mechanisms to notify DOTS
clients when conflicts occur, including the conflict cause
(Section 4.4).

In deployments where one or more translators (e.g., Traditional NAT
[RFC3022], CGN [RFC6888], NAT64 [RFC6146], NPTv6 [RFC6296]) are
enabled between the client's network and the DOTS server, DOTS signal
channel messages forwarded to a DOTS server must not include internal
IP addresses/prefixes and/or port numbers; external addresses/
prefixes and/or port numbers as assigned by the translator must be
used instead. This document does not make any recommendation about
possible translator discovery mechanisms. The following are some
(non-exhaustive) deployment examples that may be considered:

o Port Control Protocol (PCP) [RFC6887] or Session Traversal
   Utilities for NAT (STUN) [RFC5389] may be used to retrieve the
   external addresses/prefixes and/or port numbers. Information
   retrieved by means of PCP or STUN will be used to feed the DOTS
   signal channel messages that will be sent to a DOTS server.

o A DOTS gateway may be co-located with the translator. The DOTS
   gateway will need to update the DOTS messages, based upon the
   local translator's binding table.

4. DOTS Signal Channel: Messages & Behaviors

4.1. DOTS Server(s) Discovery

   This document assumes that DOTS clients are provisioned with the
   reachability information of their DOTS server(s) using a variety of
   means (e.g., local configuration, or dynamic means such as DHCP).
   These means are out of scope of this document.

   Likewise, it is out of scope of this document to specify the behavior
   of a DOTS client when it sends requests (e.g., contact all servers,
   select one server among the list) when multiple DOTS servers are
   provisioned.

4.2. CoAP URIs

   The DOTS server MUST support the use of the path-prefix of "/.well-
   known/" as defined in [RFC5785] and the registered name of "dots".
   Each DOTS operation is indicated by a path-suffix that indicates the
   intended operation. The operation path (Table 1) is appended to the
   path-prefix to form the URI used with a CoAP request to perform the
   desired DOTS operation.

Operation	Operation path	Details
Mitigation	/v1/mitigate	Section 4.4
Session configuration	/v1/config	Section 4.5

           Table 1: Operations and their corresponding URIs

4.3. Happy Eyeballs for DOTS Signal Channel

   [I-D.ietf-dots-requirements] mentions that DOTS agents will have to
   support both connectionless and connection-oriented protocols. As
   such, the DOTS signal channel is designed to operate with DTLS over
   UDP and TLS over TCP. Further, a DOTS client may acquire a list of

IPv4 and IPv6 addresses (Section 4.1), each of which can be used to
contact the DOTS server using UDP and TCP. The following specifies
the procedure to follow to select the address family and the
transport protocol for sending DOTS signal channel messages.

Such procedure is needed to avoid experiencing long connection
delays. For example, if an IPv4 path to reach a DOTS server is
found, but the DOTS server's IPv6 path is not working, a dual-stack
DOTS client may experience a significant connection delay compared to
an IPv4-only DOTS client. The other problem is that if a middlebox
between the DOTS client and DOTS server is configured to block UDP
traffic, the DOTS client will fail to establish a DTLS session with
the DOTS server and, as a consequence, will have to fall back to TLS
over TCP, thereby incurring significant connection delays.

To overcome these connection setup problems, the DOTS client attempts
to connect to its DOTS server(s) using both IPv6 and IPv4, and tries
both DTLS over UDP and TLS over TCP in a manner similar to the Happy
Eyeballs mechanism [RFC6555]. These connection attempts are
performed by the DOTS client when it initializes. The results of the
Happy Eyeballs procedure are used by the DOTS client for sending its
subsequent messages to the DOTS server.

The order of preference of DOTS signal channel address family and
transport protocol (most preferred first) is: UDP over IPv6, UDP over
IPv4, TCP over IPv6, and finally TCP over IPv4. This order adheres
to the address preference order specified in [RFC6724] and the DOTS
signal channel preference which privileges the use of UDP over TCP
(to avoid TCP's head of line blocking).

In reference to Figure 4, the DOTS client sends two TCP SYNs and two
DTLS ClientHello messages at the same time over IPv6 and IPv4. In
this example, it is assumed that the IPv6 path is broken and UDP
traffic is dropped by a middlebox but has little impact to the DOTS
client because there is no long delay before using IPv4 and TCP. The
DOTS client repeats the mechanism to discover whether DOTS signal
channel messages with DTLS over UDP becomes available from the DOTS
server, so the DOTS client can migrate the DOTS signal channel from
TCP to UDP. Such probing SHOULD NOT be done more frequently than
every 24 hours and MUST NOT be done more frequently than every 5
minutes.

```
+----------+                               +----------+
|DOTS client|                              |DOTS server|
+----------+                               +----------+
     |                                          |
     |--DTLS ClientHello, IPv6 ---->X           |
     |--TCP SYN, IPv6-------------->X           |
     |--DTLS ClientHello, IPv4 ---->X           |
     |--TCP SYN, IPv4----------------------------------->|
     |--DTLS ClientHello, IPv6 ---->X           |
     |--TCP SYN, IPv6-------------->X           |
     |<-TCP SYNACK---------------------------------------|
     |--DTLS ClientHello, IPv4 ---->X           |
     |--TCP ACK------------------------------------------>|
     |<------------Establish TLS Session-------------------->|
     |---------------DOTS signal-------------------------->|
     |                                          |
```
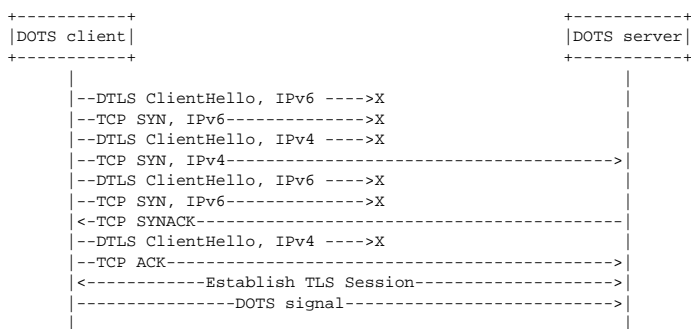
                    Figure 4: DOTS Happy Eyeballs

4.4. DOTS Mitigation Methods

   The following methods are used by a DOTS client to request, withdraw,
   or retrieve the status of mitigation requests:

   PUT: DOTS clients use the PUT method to request mitigation from a
           DOTS server (Section 4.4.1). During active mitigation, DOTS
           clients may use PUT requests to carry mitigation efficacy
           updates to the DOTS server (Section 4.4.3).

   GET: DOTS clients may use the GET method to subscribe to DOTS
           server status messages, or to retrieve the list of its
           mitigations maintained by a DOTS server (Section 4.4.2).

   DELETE: DOTS clients use the DELETE method to withdraw a request for
           mitigation from a DOTS server (Section 4.4.4).

   Mitigation request and response messages are marked as Non-
   confirmable messages (Section 2.2 of [RFC7252]).

   DOTS agents SHOULD follow the data transmission guidelines discussed
   in Section 3.1.3 of [RFC8085] and control transmission behavior by
   not sending more than one UDP datagram per RTT to the peer DOTS agent
   on average.

   Requests marked by the DOTS client as Non-confirmable messages are
   sent at regular intervals until a response is received from the DOTS
   server. If the DOTS client cannot maintain an RTT estimate, it
   SHOULD NOT send more than one Non-confirmable request every 3
   seconds, and SHOULD use an even less aggressive rate whenever
   possible (case 2 in Section 3.1.3 of [RFC8085]).

4.4.1. Request Mitigation

   When a DOTS client requires mitigation for some reason, the DOTS
   client uses the CoAP PUT method to send a mitigation request to its
   DOTS server(s) (Figure 5, illustrated in JSON diagnostic notation).

   If this DOTS client is entitled to solicit the DOTS service, the DOTS

server can enable mitigation on behalf of the DOTS client by
communicating the DOTS client's request to the mitigator and relaying
selected mitigator feedback to the requesting DOTS client.

```
   Header: PUT (Code=0.03)
   Uri-Host: "host"
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "version"
   Uri-Path: "mitigate"
   Uri-Query: "cuid=xyz"
   Content-Type: "application/cbor"
   {
     "mitigation-scope": {
       "client identifier": [
             "string"
        ],
       "scope": [
         {
           "mitigation-id": integer,
           "target-prefix": [
              "string"
            ],
           "target-port-range": [
              {
                "lower-port": integer,
                "upper-port": integer
              }
            ],
            "target-protocol": [
              integer
            ],
            "target-fqdn": [
              "string"
            ],
            "target-uri": [
              "string"
            ],
            "alias-name": [
              "string"
            ],
            "lifetime": integer
         }
        ]
     }
   }
```

                 Figure 5: PUT to convey DOTS mitigation requests

   The parameters are described below:

     client-identifier: The client

     cuid: Stands for Client Unique Identifier. A unique identifier MAY be conveyed by a
          server-side DOTS gateway that
          is meant to propagate the prevent collisions among DOTS client identity
          from the gateway's client-side to the gateway's server-side, and clients from the gateway's server-side to the DOTS server. 'client-
          identifier' MAY same
          domain. It MUST be used generated by the final DOTS server for policy
          enforcement purposes.

          The 'client-identifier' value MUST clients. A variety of
          methods can be assigned by used to generate such identifier, e.g.,
          cryptographic means [RFC4086], mimic the server-side
          DOTS gateway algorithm in [RFC4941],
          prepend a manner that ensures that there is zero
          probability that the same value will be assigned timestamp to a different
          DOTS client. randomly generated identifier, etc. The server-side DOTS gateway MUST conceal
          potentially sensitive DOTS client identity information.

          If aggregating DOTS mitigation requests received from multiple
          DOTS clients is enabled, the server side DOTS gateway has to
          include a list of 'client-identifier' values; each value
          CUID is
          pointing intended to be stable when communicating with a unique given DOTS client that is in
          server, i.e., the aggregated list.
          It is out of scope of this document to specify how aggregation is
          implemented CUID used by a DOTS gateway.

          The client-identifier attribute MUST client SHOULD NOT change
          over time. Distinct CUIDs MAY be generated and included
          by used per DOTS clients. server.

          DOTS servers MUST ignore client-identifier attributes that are
          directly supplied by source DOTS clients. This implies that first
          server-side DOTS gateways treat CUIDs as opaque values and MUST strip client-identifier attributes
          supplied by only
          compare CUIDs for equality. That is, DOTS clients. servers must not
          interpret CUIDs. DOTS servers MAY support MUST return 4.16 (CUID Collision)
          error code to a
          configuration parameter DOTS peer to identify notify that the CUID is already in-
          use by another DOTS gateways client of the same domain. Upon receipt of
          that error code, a new CUID MUST be generated by the DOTS peer.

          Client-domain DOTS gateways MAY rewrite the CUIDs used by internal
          DOTS clients. Triggers for such rewriting are trusted
          to supply client-identifier attributes. out of scope.

          This is an optional a mandatory attribute.

mitigation-id: Identifier for the mitigation request represented
   with an integer. This identifier MUST be unique for each
   mitigation request bound to the DOTS client, i.e., the
   'mitigation-id' parameter value in the mitigation request needs to
   be unique relative to the 'mitigation-id' parameter values of
   active mitigation requests conveyed from the DOTS client to the
   DOTS server. This identifier MUST be generated by the DOTS
   client. This document does not make any assumption about how this
   identifier is generated.

   This is a mandatory attribute.

target-prefix: A list of prefixes identifying resources under
   attack. Prefixes are represented using Classless Inter-Domain
   Routing (CIDR) notation [RFC4632].
   As a reminder, the prefix length must be less than or equal to 32
   (resp. 128) for IPv4 (resp. IPv6).

   This is an optional attribute.

target-port-range: A list of port numbers bound to resources under
   attack.

   The port range is defined by two bounds, a lower port number
   (lower-port) and an upper port number (upper-port). When only
   'lower-port' is present, it represents a single port number. For
   TCP, UDP, Stream Control Transmission Protocol (SCTP) [RFC4960],
   or Datagram Congestion Control Protocol (DCCP) [RFC4340], the
   range of ports can be, for example, 1024-65535.

   This is an optional attribute.

target-protocol: A list of protocols involved in an attack. Values
   are taken from the IANA protocol registry [proto_numbers].

   The value '0' has a special meaning for 'all protocols'.

   This is an optional attribute.

target-fqdn: A list of Fully Qualified Domain Names (FQDNs)
   identifying resources under attack. An FQDN is the full name of a
   resource, rather than just its hostname. For example, "venera" is
   a hostname, and "venera.isi.edu" is an FQDN.

   This is an optional attribute.

target-uri: A list of Uniform Resource Identifiers (URIs) [RFC3986]
   identifying resources under attack.

   This is an optional attribute.

alias-name: A list of aliases of resources for which the mitigation
   is requested. Aliases can be created using the DOTS data channel
   (Section 6.1 of [I-D.ietf-dots-data-channel]), direct
   configuration, or other means. An alias is used in subsequent
   signal channel exchanges to refer more efficiently to the
   resources under attack.

   This is an optional attribute.

lifetime: Lifetime of the mitigation request in seconds. The
   RECOMMENDED lifetime of a mitigation request is 3600 seconds (60
   minutes) -- this value was chosen to be long enough so that
   refreshing is not typically a burden on the DOTS client, while
   expiring the request where the client has unexpectedly quit in a
   timely manner. DOTS clients MUST include this parameter in their
   mitigation requests. Upon the expiry of this lifetime, and if the
   request is not refreshed, the mitigation request is removed. The
   request can be refreshed by sending the same request again.

   A lifetime of ~~0~~ **'0'** in a mitigation request is an invalid value.

   A lifetime of negative one (-1) indicates indefinite lifetime for
   the mitigation request. The DOTS server MAY refuse indefinite
   lifetime, for policy reasons; the ~~granted lifetime value~~ **granted lifetime value is
   returned in the response. DOTS clients MUST be prepared to not be
   granted mitigations with indefinite lifetimes.**

   **The DOTS server MUST always indicate the actual lifetime in the
   response and the remaining lifetime in status messages sent to the
   DOTS client.**

   This is a mandatory attribute.

**In deployments where server-domain DOTS gateways are enabled, some
information about the origin source client domain has to be supplied
to the DOTS server. That information is meant to assist the DOTS
server to enforce some policies. Figure 6 shows an example of a
request relayed by a server-domain DOTS gateway.**

```
  Header: PUT (Code=0.03)
  Uri-Host: "host"
  Uri-Path: ".well-known"
  Uri-Path: "dots"
  Uri-Path: "version"
  Uri-Path: "mitigate"
  Uri-Query: "cuid=xyz"
  Content-Type: "application/cbor"
  {
    "mitigation-scope": {
```

```
          "client-domain-hash": "string",
          "scope": [
            {
              "mitigation-id": integer,
              "target-prefix": [
                "string"
              ],
              "target-port-range": [
                {
                  "lower-port": integer,
                  "upper-port": integer
                }
              ],
              "target-protocol": [
                integer
              ],
              "target-fqdn": [
                "string"
              ],
              "target-uri": [
                "string"
              ],
              "alias-name": [
                "string"
              ],
              "lifetime": integer
            }
          ]
        }
      }
```

          Figure 6: PUT to convey DOTS mitigation request as relayed by a
                            server-side DOTS gateway

   The DOTS gateway may add the following parameters:

   client-domain-hash: The client identifier MAY be conveyed by a
      server-domain DOTS gateway to propagate the source domain identity
      from the gateway's client-side to the gateway's server-side, and
      from the gateway's server-side to the DOTS server. 'client-domain-
      hash' MAY be used by the final DOTS server for policy enforcement
      purposes (e.g., enforce a quota on filtering rules).

      The 'client-domain-hash' value MUST be assigned by the server-
      domain DOTS gateway in a manner that ensures that there is zero
      probability that the same value will be assigned to a different
      client domain.

      If the DOTS client is using the certificate provisioned by the
      Enrollment over Secure Transport (EST) server [RFC7030] in the
      DOTS gateway-domain to authenticate itself to the DOTS gateway,
      the 'client-domain-hash' value may be the output of a
      cryptographic hash algorithm whose input is the DER-encoded ASN.1
      representation of the Subject Public Key Info (SPKI) of an X.509
      certificate. In this version of the specification, the
      cryptographic hash algorithm used is SHA-256 [RFC6234]. The
      output of the cryptographic hash algorithm is truncated to 16
      bytes; truncation is done by stripping off the final 16 bytes.
      The truncated output is base64url encoded.

      The 'client-domain-hash' attribute MUST NOT be generated and
      included by DOTS clients.

      DOTS servers MUST ignore 'client-domain-hash' attributes that are
      directly supplied by source DOTS clients or client-domain DOTS
      gateways. This implies that first server-domain DOTS gateways
      MUST strip 'client-domain-hash' attributes supplied by DOTS
      clients. DOTS servers MAY support a configuration parameter to
      identify DOTS gateways that are trusted to supply 'client-domain-
      hash' attributes.

      Only singe-valued 'client-domain-hash' are defined in this
      document.

      This is an optional attribute.

   hop-limit: This attribute is used to detect and prevent infinite
      loops. This attribute is typically inserted by a DOTS gateway.

      Each intermediate DOTS agent involved in the handling of a DOTS
      message MUST decrement the hop-limit value by 1 prior to
      forwarding upstream if this parameter exists. DOTS messages MUST
      NOT be forwarded if the value of hop-limit is set to '0' after
      decrement. Messages that cannot be forwarded because of exhausted
      hop-limit SHOULD be logged with a 5.06 (Hop Limit Reached) error
      message sent back to the DOTS peer. It is
      returned in the response. RECOMMENDED that DOTS
      clients MUST be prepared and gateways support means to not be
      granted mitigations with indefinite lifetimes. alert administrators about
      loop errors so that appropriate actions are undertaken.

      The DOTS server MUST always indicate initial hop-limit value SHOULD be configurable. If no initial
      value is explicitly provided, the actual lifetime in default initial hop-limit value
      of 16 MUST be used.

      Because forwarding errors may occur if inadequate hop-limit values
      are used, DOTS agents at the
      response and boundaries of an administrative
      domain MAY be instructed to rewrite the remaining lifetime value of hop-limit carried
      in status received messages sent to (that is, ignore the
```

DOTS client. **value of hop-limit received in a message).**

This is a mandatory **an optional** attribute.

Because of the complexity to handle partial failure cases, this specification does not allow for including multiple mitigation requests in the same PUT request.  Concretely, a DOTS client MUST NOT include multiple 'scope' parameters in the same PUT request.

The CBOR key values for the parameters are defined in Section 6. Section 9 defines how the CBOR key values can be allocated to standard bodies and vendors.

FQDN and URI mitigation scopes may be thought of as a form of scope alias, in which the addresses to which the domain name or URI resolve represent the full scope of the mitigation.

In the PUT request at least one of the attributes 'target-prefix' or 'target-fqdn' or 'target-uri 'or 'alias-name' MUST be present. Attributes with emty **empty** values MUST NOT be present in a request.

The relative order of two mitigation requests from a DOTS client is determined by comparing their respective 'mitigation-id' values.  If two mitigation requests have overlapping mitigation scopes, the mitigation request with the highest numeric 'mitigation-id' value will override the other mitigation request.  Two mitigation-ids from a DOTS client are overlapping if there is a common IP address, IP prefix, FQDN, URI, or alias-name.  To avoid maintaining a long list of overlapping mitigation requests from a DOTS client and avoid error-prone provisioning of mitigation requests from a DOTS client, the overlapped lower numeric 'mitigation-id' MUST be automatically deleted and no longer available at the DOTS server.

The Uri-Path option carries a major and minor version nomenclature to manage versioning and DOTS signal channel in this specification uses v1 major version.

Figure 6 **7** shows a PUT request example to signal that ports 80, 8080, and 443 used by 2001:db8:6401::1 and 2001:db8:6401::2 servers are under attack (illustrated in JSON diagnostic notation).  **The presence of 'client-domain-hash' indicates that a server-domain DOTS gateway has modified the initial PUT request sent by the DOTS client.**

```
  Header: PUT (Code=0.03)
  Uri-Host: "www.example.com"
  Uri-Path: ".well-known"
  Uri-Path: "dots"
  Uri-Path: "v1"
  Uri-Path: "mitigate"
  Uri-Query: "cuid=xyz"
  Content-Format: "application/cbor"
  {
    "mitigation-scope": {
      "client-identifier": [
          "dz6pHjaADkaFTbjr0JGBpw"
        ],
      "client-domain-hash": "dz6pHjaADkaFTbjr0JGBpw",
      "scope": [
        {
          "mitigation-id": 12332,
          "target-prefix": [
             "2001:db8:6401::1/128",
             "2001:db8:6401::2/128"
           ],
          "target-port-range": [
            {
              "lower-port": 80
            },
            {
              "lower-port": 443
            },
            {
               "lower-port": 8080
            }
            ],
          "target-protocol": [
             6
            ]
        }
      ]
    }
  }
```

Figure 6: **7:** PUT for DOTS signal **mitigation request**

The corresponding CBOR encoding format is shown in Figure 7. **8.**

```
A1                                  # map(1)
   01                               # unsigned(1)
   A2                               # map(2)
      18 20                         # unsigned(32)
         81 24                      # array(1) unsigned(36)
         76                         # text(22)
            647A3670486A6141446B614654626A72304A47427077 # "dz6pHjaADkaFTbjr0JGBpw"
      02                            # unsigned(2)
      81                            # array(1)
         A4                         # map(4)
            03                      # unsigned(3)
            19 302C                 # unsigned(12332)
```

```
                04
                18 23                   # unsigned(4) unsigned(35)
                82                      # array(2)
                   74                   # text(20)
                      323030313A6462383A363430313A3A312F313238 # "2001:db8:6401::1/128"
                   74                   # text(20)
                      323030313A6462383A363430313A3A322F313238 # "2001:db8:6401::2/128"
                05                      # unsigned(5)
                83                      # array(3)
                   A1                   # map(1)
                      06                # unsigned(6)
                      18 50             # unsigned(80)
                   A1                   # map(1)
                      06                # unsigned(6)
                      19 01BB           # unsigned(443)
                   A1                   # map(1)
                      06                # unsigned(6)
                      19 1F90           # unsigned(8080)
                08                      # unsigned(8)
                81                      # array(1)
                   06                   # unsigned(6)
```

                    Figure 7+ **8:** PUT for DOTS signal (CBOR)

   ~~If the DOTS client is using the certificate provisioned by the
   Enrollment over Secure Transport (EST) server [RFC7030] in the DOTS
   gateway domain to authenticate itself to the DOTS gateway, then the
   'client-identifier' value can be the output of a cryptographic hash
   algorithm whose input is the DER-encoded ASN.1 representation of the
   Subject Public Key Info (SPKI) of an X.509 certificate.~~

   ~~In this version of the specification, the cryptographic hash
   algorithm used is SHA-256 [RFC6234]. The output of the cryptographic
   hash algorithm is truncated to 16 bytes; truncation is done by
   stripping off the final 16 bytes. The truncated output is base64url
   encoded.~~

   In both DOTS signal and data channel sessions, the DOTS client MUST
   authenticate itself to the DOTS server (Section 8). The DOTS server
   may use the algorithm presented in Section 7 of [RFC7589] to derive
   the DOTS client identity or username from the client certificate.
   The DOTS client identity allows the DOTS server to accept mitigation
   requests with scopes that the DOTS client is authorized to manage.
   The DOTS server couples the DOTS signal and data channel sessions
   using the DOTS client identity ~~and~~ **or** the ~~'client-identifier'~~ **'client-domain-hash'** parameter
   value, so the DOTS server can validate whether the aliases conveyed
   in the mitigation request were indeed created by the same DOTS client
   using the DOTS data channel session. If the aliases were not created
   by the DOTS client, the DOTS server ~~returns~~ **MUST return** 4.00 (Bad Request) in
   the response.

   The DOTS server couples the DOTS signal channel sessions using the
   DOTS client identity ~~and~~ **or** the ~~'client-identifier'~~ **'client-domain-hash'** parameter value, and
   the DOTS server uses 'mitigation-id' **and 'cuid'** parameter ~~value~~ **values** to
   detect duplicate mitigation requests. If the mitigation request
   contains the alias-name and other parameters identifying the target
   resources (such as, 'target-prefix', 'target-port-range', ~~'target-fqdn',~~ **'target-
   fqdn',** or 'target-uri'), ~~then~~ the DOTS server appends the parameter values
   in 'alias-name' with the corresponding parameter values in 'target-
   prefix', 'target-port-range', 'target-fqdn', or 'target-uri'.

   The DOTS server indicates the result of processing the PUT request
   using CoAP response codes. CoAP 2.xx codes are success. CoAP 4.xx
   codes are some sort of invalid requests (client errors). COAP 5.xx
   codes are returned if the DOTS server has erred or is currently
   unavailable to provide mitigation in response to the mitigation
   request from the DOTS client.

   Figure ~~8~~ **9** shows an example of a PUT request that is successfully
   processed **by a DOTS server** (i.e., CoAP 2.xx response codes).

```
   {
     "mitigation-scope": {
       "client-identifier": [
            "string"
       ],
       "client-domain-hash": "string",
       "scope": [
          {
            "mitigation-id": 12332,
            "lifetime": 3600
          }
       ]
     }
   }
```

                      Figure ~~8+~~ **9:** 2.xx response body

   If the request is missing one or more mandatory attributes, or
   includes multiple 'scope' parameters, or contains invalid or unknown
   parameters, the DOTS server MUST reply with 4.00 (Bad Request). DOTS
   agents can safely ignore Vendor-Specific parameters they don't
   understand.

   A DOTS server that receives a mitigation request with a lifetime set
   to '0' MUST reply with a 4.00 (Bad Request).

   If the DOTS server does not find the 'mitigation-id' parameter value
   conveyed in the PUT request in its configuration data, it MAY accept
   the mitigation request by sending back a 2.01 (Created) response to

```
      the DOTS client; the DOTS server will consequently try to mitigate
      the attack.

      If the DOTS server finds the 'mitigation-id' parameter value conveyed
      in the PUT request in its configuration data, data bound to that DOTS
      client, it MAY update the mitigation request, and a 2.04 (Changed)
      response is returned to indicate a successful update of the
      mitigation request.

      If the request is conflicting with an existing mitigation request
      from a different DOTS client, and the DOTS server decides to maintain
      the conflicting mitigation request, the DOTS server returns 4.09
      (Conflict) [RFC8132] to the requesting DOTS client.  The response
      includes enough information for a DOTS client to recognize the source
      of the conflict (refer to 'conflict-information' specified in
      Section 4.4.2).

      For a mitigation request to continue beyond the initial negotiated
      lifetime, the DOTS client has to refresh the current mitigation
      request by sending a new PUT request.  This PUT request MUST use the
      same 'mitigation-id' value, and MUST repeat all the other parameters
      as sent in the original mitigation request apart from a possible
      change to the lifetime parameter value.

      The DOTS gateway, which inserted a 'client-identifier' 'client-domain-hash' attribute in
      a request, MUST strip the 'client-identifier' 'client-domain-hash' parameter in the
      corresponding response before forwarding the response to the DOTS
      client.  If we consider the example depicted in Figure 9, the message
      that will be relayed by the DOTS gateway is shown in Figure 10.

      {
        "mitigation-scope": {
          "scope": [
              {
                "mitigation-id": 12332,
                "lifetime": 3600
              }
          ]
        }
      }

            Figure 10: 2.xx response body relayed by a DOTS gateway

   4.4.2.  Retrieve Information Related to a Mitigation

      A GET request is used by a DOTS client to retrieve information
      (including status) of DOTS mitigations from a DOTS server.

      The same considerations for manipulating 'client-identifier' 'client-domain-hash'
      parameter by a server-domain DOTS gateway gateways specified in Section 4.4.1
      MUST be followed for GET requests.

      If the DOTS server does not find the 'mitigation-id' parameter value
      conveyed in the GET request in its configuration data for the
      requesting DOTS client or the one identified by 'client-identifier', 'client-domain-hash',
      it MUST respond with a 4.04 (Not Found) error response code.
      Likewise, the same error MUST be returned as a response to a request
      to retrieve all mitigation records of a given DOTS client if the DOTS
      server does not find any mitigation record for that DOTS client or
      the one identified by 'client-identifier'. 'client-domain-hash'.

      The 'c' (content) parameter and its permitted values defined in
      [I-D.ietf-core-comi] can be used to retrieve non-configuration data
      (attack mitigation status) or configuration data or both.  The DOTS
      server may support this optional filtering capability.  It can safely
      ignore it if not supported.

      The following examples illustrate how a DOTS client retrieves active
      mitigation requests from a DOTS server.  In particular:

      o  Figure 9 11 shows the example of a GET request to retrieve all DOTS
         mitigation requests signaled by a DOTS client.

      o  Figure 10 12 shows the example of a GET request to retrieve a
         specific DOTS mitigation request signaled by a DOTS client.  The
         configuration data to be reported in the response is formatted in
         the same order it was processed by the DOTS server.

      These two examples assume the default of "c=a"; that is, the DOTS
      client asks for all data to be reported by the DOTS server.

        Header: GET (Code=0.01)
        Uri-Host: "host"
        Uri-Path: ".well-known"
        Uri-Path: "dots"
        Uri-Path: "version"
        Uri-Path: "mitigate"
        Uri-Query: "cuid=xyz"
        Observe : 0
        {
            "mitigation-scope": {
              "client-identifier": {
                "dz6pHjaADkaFTbjr0JGBpw"
              }
            }
        }

             Figure 9: 11: GET to retrieve all DOTS mitigation requests

        Header: GET (Code=0.01)
```

```
      Uri-Host: "host"
      Uri-Path: ".well-known"
      Uri-Path: "dots"
      Uri-Path: "version"
      Uri-Path: "mitigate"
      Uri-Query: "cuid=xyz"
      Uri-Query: "mitigation-id=12332"
      Observe : 0
      Content-Format: "application/cbor"
      {
        "mitigation-scope": {
          "client-identifier": [
             "dz6pHjaADkaFTbjr0JGBpw"
          ],
          "scope": [
             {
               "mitigation-id": 12332
             }
          ]
        }
      }
```

                Figure 10+ **12:** GET to retrieve a specific DOTS mitigation request

     Figure 11 **13** shows a response example of all active mitigation requests
     associated with the DOTS client on the DOTS server and the mitigation
     status of each mitigation request.

```
    {
      "mitigation-scope": {
        "scope": [
          {
            "mitigation-id": 12332,
            "mitigation-start": 1507818434.00,
            "target-prefix": [
                "2001:db8:6401::1/128",
                "2001:db8:6401::2/128"
            ],
            "target-protocol": [
              17
            ],
            "lifetime": 1800,
            "status": 2,
            "bytes-dropped": 134334555,
            "bps-dropped": 43344,
            "pkts-dropped": 333334444,
            "pps-dropped": 432432
          },
          {
            "mitigation-id": 12333,
            "mitigation-start": 1507818393.00,
            "target-prefix": [
                "2001:db8:6401::1/128",
                "2001:db8:6401::2/128"
            ],
            "target-protocol": [
              6
            ],
            "lifetime": 1800,
            "status": 3,
            "bytes-dropped": 0,
            "bps-dropped": 0,
            "pkts-dropped": 0,
            "pps-dropped": 0
          }
        ]
      }
    }
```

                          Figure 11+ **13:** Response body

     The mitigation status parameters are described below:

     mitigation-start: Mitigation start time is expressed in seconds
        relative to 1970-01-01T00:00Z in UTC time (Section 2.4.1 of

        [RFC7049]). The encoding is modified so that the leading tag 1
        (epoch-based date/time) MUST be omitted.

        This is a mandatory attribute.

     lifetime: The remaining lifetime of the mitigation request, in
        seconds.

        This is a mandatory attribute.

     status: Status of attack mitigation. The various possible values of
        'status' parameter are explained in Table 2.

        This is a mandatory attribute.

     conflict-information: Indicates that a mitigation request is
        conflicting with another mitigation request(s) from other DOTS
        client(s). This optional attribute has the following structure:

        conflict-status: Indicates the status of a conflicting mitigation
           request. The following values are defined:

           1: DOTS server has detected conflicting mitigation requests
                from different DOTS clients. This mitigation request is

```
                 currently inactive until the conflicts are resolved.
                 Another mitigation request is active.

            2:  DOTS server has detected conflicting mitigation requests
                from different DOTS clients.  This mitigation request is
                currently active.

            3:  DOTS server has detected conflicting mitigation requests
                from different DOTS clients.  All conflicting mitigation
                requests are inactive.

         conflict-cause:  Indicates the cause of the conflict.  The
            following values are defined:

            1:  Overlapping targets. 'conflict-scope' provides more details
                about the conflicting target clauses.

            2:  Conflicts with an existing white list.  This code is
                returned when the DDoS mitigation detects source addresses/
                prefixes in the white-listed ACLs are attacking the target.

         conflict-scope  Indicates the conflict scope.  It may include a
            list of IP addresses, a list of prefixes, a list of port
            numbers, a list of target protocols, a list of FQDNs, a list of
            URIs, a list of alias-names, or references to conflicting ACLs.

         retry-timer  Indicates, in seconds, the time after which the DOTS
            client may re-issue the same request.  The DOTS server returns
            'retry-timer' only to DOTS client(s) for which a mitigation
            request is deactivated.  Any retransmission of the same
            mitigation request before the expiry of this timer is likely to
            be rejected by the DOTS server for the same reasons.

            The retry-timer SHOULD be equal to the lifetime of the active
            mitigation request resulting in the deactivation of the
            conflicting mitigation request.  The lifetime of the
            deactivated mitigation request will be updated to (retry-timer
            + 45 seconds), so the DOTS client can refresh the deactivated
            mitigation request after retry-timer seconds before expiry of
            lifetime and check if the conflict is resolved.

   bytes-dropped:  The total dropped byte count for the mitigation
      request since the attack mitigation is triggered.  The count wraps
      around when it reaches the maximum value of unsigned integer.

      This is an optional attribute.

   bps-dropped:  The average number of dropped bytes per second for the
      mitigation request since the attack mitigation is triggered.  This
      SHOULD be a five-minute average.

      This is an optional attribute.

   pkts-dropped:  The total number of dropped packet count for the
      mitigation request since the attack mitigation is triggered.

      This is an optional attribute.

   pps-dropped:  The average number of dropped packets per second for
      the mitigation request since the attack mitigation is triggered.
      This SHOULD be a five-minute average.

      This is an optional attribute.
```

Parameter value	Description
1	Attack mitigation is in progress (e.g., changing the network path to re-route the inbound traffic to DOTS mitigator).
2	Attack is successfully mitigated (e.g., traffic is redirected to a ~~DDOS~~ **DDoS** mitigator and attack traffic is dropped).
3	Attack has stopped and the DOTS client can withdraw the mitigation request.
4	Attack has exceeded the mitigation provider capability.
5	DOTS client has withdrawn the mitigation request and the mitigation is active but terminating.
6	Attack mitigation is now terminated.
7	Attack mitigation is withdrawn.
8	Attack mitigation is rejected.

                 Table 2: Values of 'status' parameter

```
   The observe option defined in [RFC7641] extends the CoAP core
   protocol with a mechanism for a CoAP client to "observe" a resource
   on a CoAP server: The client retrieves a representation of the
   resource and requests this representation be updated by the server as
   long as the client is interested in the resource.  A DOTS client
   conveys the observe option set to '0' in the GET request to receive
```

unsolicited notifications of attack mitigation status from the DOTS
server.

Unidirectional notifications within the bidirectional signal channel
allows unsolicited message delivery, enabling asynchronous
notifications between the agents. Due to the higher likelihood of
packet loss during a DDoS attack, the DOTS server periodically sends
attack mitigation status to the DOTS client and also notifies the
DOTS client whenever the status of the attack mitigation changes. If
the DOTS server cannot maintain a RTT estimate, it SHOULD NOT send
more than one unsolicited notification every 3 seconds, and SHOULD
use an even less aggressive rate whenever possible (case 2 in
Section 3.1.3 of [RFC8085]). **The DOTS server MUST use the same CUID
as the one used by the DOTS client to observe a mitigation request.**

When conflicting requests are detected, the DOTS server enforces the
corresponding policy (e.g., accept all requests, reject all requests,
accept only one request but reject all the others, ...). It is
assumed that this policy is supplied by the DOTS server administrator
or it is a default behavior of the DOTS server implementation. Then,
the DOTS server sends notification message(s) to the DOTS client(s)
at the origin of the conflict. A conflict notification message
includes information about the conflict cause, scope, and the status
of the mitigation request(s). For example,

o A notification message with status code set to '8 (Attack
   mitigation is rejected)' and 'conflict-status' set to '1' is sent
   to a DOTS client to indicate that this mitigation request is
   rejected because a conflict is detected.

o A notification message with status code set to '7 (Attack
   mitigation is withdrawn)' and 'conflict-status' set to '1' is sent
   to a DOTS client to indicate that an active mitigation request is
   deactivated because a conflict is detected.

o A notification message with status code set to '1 (Attack
   mitigation is in progress)' and 'conflict-status' set to '2' is
   sent to a DOTS client to indicate that this mitigation request is
   in progress, but a conflict is detected.

Upon receipt of a conflict notification message indicating that a
mitigation request is deactivated because of a conflict, a DOTS
client MUST NOT resend the same mitigation request before the expiry
of 'retry-timer'. It is also recommended that DOTS clients support
means to alert administrators about mitigation conflicts.

A DOTS client that is no longer interested in receiving notifications
from the DOTS server can simply "forget" the observation. When the
DOTS server sends the next notification, the DOTS client will not
recognize the token in the message and thus will return a Reset
message. This causes the DOTS server to remove the associated entry.
Alternatively, the DOTS client can explicitly deregister itself by
issuing a GET request that has the Token field set to the token of
the observation to be cancelled and includes an Observe Option with
the value set to '1' (deregister).

Figure ~~12~~ **14** shows an example of a DOTS client requesting a DOTS server
to send notifications related to a given mitigation request.

```
       DOTS Client               DOTS Server

   +-----------+             +-----------+
   |DOTS client|             |DOTS server|
   +-----------+             +-----------+
         |                         |
         | GET /<mitigation-id number> |
         | Token: 0x4a             |      Registration
         | Observe: 0              |
         +------------------------->|
         |                         |
         | 2.05 Content            |
         | Token: 0x4a             |      Notification of
         | Observe: 12             |      the current state
         | status: "mitigation     |
         |          in progress"   |
         |<-------------------------+
         | 2.05 Content            |
         | Token: 0x4a             |      Notification upon
         | Observe: 44             |       a state change
         | status: "mitigation     |
         |          complete"      |
         |<-------------------------+
         | 2.05 Content            |
         | Token: 0x4a             |      Notification upon
         | Observe: 60             |      a state change
         | status: "attack stopped"|
         |<-------------------------+
         |                         |
```

       Figure ~~12:~~ **14:** Notifications of attack mitigation status

4.4.2.1. Mitigation Status

The DOTS client can send the GET request at frequent intervals
without the Observe option to retrieve the configuration data of the
mitigation request and non-configuration data (i.e., the attack
status). The frequency of polling the DOTS server to get the
mitigation status should follow the transmission guidelines given in
Section 3.1.3 of [RFC8085]. If the DOTS server has been able to
mitigate the attack and the attack has stopped, the DOTS server

indicates as such in the status, and the DOTS client recalls the
mitigation request by issuing a DELETE request for the ~~mitigation-id.~~ **'mitigation-
id'.**

A DOTS client SHOULD react to the status of the attack as per the
information sent by the DOTS server rather than acknowledging by
itself, using its own means, that the attack has been mitigated.
This ensures that the DOTS client does not recall a mitigation
request prematurely because it is possible that the DOTS client does
not sense the ~~DDOS~~ **DDoS** attack on its resources but the DOTS server could
be actively mitigating the attack and the attack is not completely
averted.

4.4.3. Efficacy Update from DOTS Clients

While DDoS mitigation is active, due to the likelihood of packet
loss, a DOTS client MAY periodically transmit DOTS mitigation
efficacy updates to the relevant DOTS server. A PUT request is used
to convey the mitigation efficacy update to the DOTS server.

The PUT request MUST include all the parameters used in the PUT
request to carry the DOTS mitigation request (Section 4.4.1)
unchanged apart from the lifetime parameter value. If this is not
the case, the DOTS server MUST reject the request with a 4.00 (Bad
Request).

The If-Match Option (Section 5.10.8.1 of [RFC7252]) with an empty
value is used to make the PUT request conditional on the current
existence of the mitigation request. If UDP is used as transport,
CoAP requests may arrive out-of-order. For example, the DOTS client
may send a PUT request to convey an efficacy update to the DOTS
server followed by a DELETE request to withdraw the mitigation
request, but the DELETE request arrives at the DOTS server before the
PUT request. To handle out-of-order delivery of requests, if an If-
Match option is present in the PUT request and the 'mitigation-id' in
the request matches a mitigation request from that DOTS client, then
the request is processed. If no match is found, the PUT request is
silently ignored.

An example of an efficacy update message, which includes an If-Match
option with an empty value, is depicted in Figure ~~13.~~ **15.**

```
   Header: PUT (Code=0.03)
   Uri-Host: "host"
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "version"
   Uri-Path: "mitigate"
   Uri-Query: "cuid=xyz"
   Content-Format: "application/cbor"
   If-Match:
   {
    "mitigation-scope": {
       "client-identifier": [
             "string"
       ],
      "scope": [
        {
          "mitigation-id": integer,
          "target-prefix": [
             "string"
           ],
          "target-port-range": [
             {
               "lower-port": integer,
               "upper-port": integer
             }
           ],
           "target-protocol": [
             integer
           ],
           "target-fqdn": [
             "string"
           ],
           "target-uri": [
             "string"
           ],
           "alias-name": [
             "string"
           ],
          "lifetime": integer,
          "attack-status": integer
        }
      ]
    }
  }
```

                        Figure ~~13:~~ **15:** Efficacy Update

The 'attack-status' parameter is a mandatory attribute when
performing an efficacy update. The various possible values contained
in the 'attack-status' parameter are described in Table 3.

```
+-----------+--------------------------------------------------------+
| Parameter | Description                                            |
|     value |                                                        |
+-----------+--------------------------------------------------------+
|         1 | The DOTS client determines that it is still under      |
|           | attack.                                                |
+-----------+--------------------------------------------------------+
```

```
|         2 | The DOTS client determines that the attack is      |
|           | successfully mitigated (e.g., attack traffic is not |
|           | seen).                                             |
+-----------+----------------------------------------------------+
```

                Table 3: Values of 'attack-status' parameter

   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 mitigation efficacy
   update. The error response code 5.03 (Service Unavailable) is
   returned if the DOTS server has erred or is incapable of performing
   the mitigation.

4.4.4. Withdraw a Mitigation

   A DELETE request is used to withdraw a DOTS mitigation request from a
   DOTS server (Figure ~~14).~~ **16).**

   The same considerations for manipulating ~~'client-identifier'~~ **'client-domain-hash'**
   parameter by ~~a~~ DOTS ~~gateway,~~ **gateways,** as specified in Section 4.4.1, MUST be
   followed for DELETE requests.

     Header: DELETE (Code=0.04)
     Uri-Host: "host"
     Uri-Path: ".well-known"
     Uri-Path: "dots"
     Uri-Path: "version"
     Uri-Path: "mitigate"
     ~~Content-Format: "application/cbor"~~
     ~~{~~
        ~~"mitigation-scope": {~~
           ~~"client-identifier": [~~
              ~~"string"~~
           ~~],~~
           ~~"scope": [~~
              ~~{~~
                 ~~"mitigation-id": integer~~
              ~~}~~
           ~~]~~
        ~~}~~
     ~~}~~
     Uri-Query: "cuid=xyz"
     Uri-Query: "mitigation-id=123"

                   Figure ~~14:~~ **16:** Withdraw DOTS ~~signal~~ **mitigation**

   If the request does not include a 'mitigation-id' parameter, the DOTS
   server MUST reply with a 4.00 (Bad Request).

   Once the request is validated, the DOTS server immediately
   acknowledges a DOTS client's request to withdraw the DOTS signal
   using 2.02 (Deleted) response code with no response payload. A 2.02
   (Deleted) Response Code is returned even if the 'mitigation-id'
   parameter value conveyed in the DELETE request does not exist in its
   configuration data before the request.

   If the DOTS server finds the 'mitigation-id' parameter value conveyed
   in the DELETE request in its configuration data for the DOTS client,
   then to protect against route or DNS flapping caused by a DOTS client
   rapidly removing a mitigation, and to dampen the effect of
   oscillating attacks, the DOTS server MAY allow mitigation to continue
   for a limited period after acknowledging a DOTS client's withdrawal
   of a mitigation request. During this period, the DOTS server status
   messages SHOULD indicate that mitigation is active but terminating
   (Section 4.4.2).

   The initial active-but-terminating period SHOULD be sufficiently long
   to absorb latency incurred by route propagation. The active-but-
   terminating period SHOULD be set by default to 120 seconds. If the
   client requests mitigation again before the initial active-but-
   terminating period elapses, the DOTS server MAY exponentially
   increase the active-but- terminating period up to a maximum of 300
   seconds (5 minutes).

   After the active-but-terminating period elapses, the DOTS server MUST
   treat the mitigation as terminated, as the DOTS client is no longer
   responsible for the mitigation. For example, if there is a financial
   relationship between the DOTS client and server domains, the DOTS
   client stops incurring cost at this point.

4.5. DOTS Signal Channel Session Configuration

   A DOTS client can negotiate, configure, and retrieve the DOTS signal
   channel session behavior. The DOTS signal channel can be used, for
   example, to configure the following:

   a. Heartbeat interval (heartbeat-interval): DOTS agents regularly
       send heartbeats (CoAP Ping/Pong) to each other after mutual
       authentication is successfully completed in order to keep the
       DOTS signal channel open. Heartbeat messages are exchanged
       between DOTS agents every 'heartbeat-interval' seconds to detect
       the current status of the DOTS signal channel session.

   b. Missing heartbeats allowed (missing-hb-allowed): This variable
       indicates the maximum number of consecutive heartbeat messages
       for which a DOTS agent did not receive a response before
       concluding that the session is disconnected or defunct.

   c. Acceptable signal loss ratio: Maximum retransmissions,

```
        retransmission timeout value, and other message transmission
        parameters for the DOTS signal channel.
```

The same or distinct configuration sets may be used during ~~attack ('attack-time-config') and peace~~ times ~~('peace-time-config').~~ **when a mitigation is active ('mitigating-config') and when no mitigation is active ('idle-config').** This is particularly useful for DOTS servers that might want to reduce heartbeat frequency or cease heartbeat exchanges when an active DOTS client has not requested mitigation. If distinct ~~configuration~~ **configurations** are used, DOTS agents MUST follow the appropriate configuration set as a function of the mitigation activity (e.g., if no mitigation request is active, ~~'peace-time-config' related~~ **'idle-config'-related** values must be followed). Additionally, DOTS agents MUST automatically switch to the other configuration upon a change in the mitigation activity (e.g., if an attack mitigation is launched after a peacetime, the DOTS agent switches from ~~'peace-time-config'~~ **'idle-config'** to ~~'attack-time-config' related~~ **'mitigating-config'-related** values).

```
    Requests and responses are deemed reliable by marking them as
    Confirmable (CON) messages.  DOTS signal channel session
    configuration requests and responses are marked as Confirmable
    messages.  As explained in Section 2.1 of [RFC7252], a Confirmable
    message is retransmitted using a default timeout and exponential
    back-off between retransmissions, until the DOTS server sends an
    Acknowledgement message (ACK) with the same Message ID conveyed from
    the DOTS client.

    Message transmission parameters are defined in Section 4.8 of
    [RFC7252].  The DOTS server can either piggyback the response in the
    acknowledgement message or, if the DOTS server cannot respond
    immediately to a request carried in a Confirmable message, it simply
    responds with an Empty Acknowledgement message so that the DOTS
    client can stop retransmitting the request.  Empty Acknowledgement
    message is explained in Section 2.2 of [RFC7252].  When the response
    is ready, the server sends it in a new Confirmable message which in
    turn needs to be acknowledged by the DOTS client (see Sections 5.2.1
    and 5.2.2 of [RFC7252]).  Requests and responses exchanged between
    DOTS agents during peacetime are marked as Confirmable messages.

      Implementation Note: A DOTS client that receives a response in a
      CON message may want to clean up the message state right after
      sending the ACK.  If that ACK is lost and the DOTS server
      retransmits the CON, the DOTS client may no longer have any state
      that would help it correlate this response, thereby unexpecting
      the retransmission message.  The DOTS client will send a Reset
      message so it does not receive any more retransmissions.  This
      behavior is normal and not an indication of an error (see
      Section 5.3.2 of [RFC7252] for more details).

4.5.1.  Discover Configuration Parameters

    A GET request is used to obtain acceptable (e.g., minimum and maximum
    values) and current configuration parameters on the DOTS server for
    DOTS signal channel session configuration.
```
This procedure occurs between a DOTS client and its immediate peer DOTS server. As such, this GET request MUST NOT be relayed by an on-path DOTS gateway.

Figure ~~15~~ **17** shows how to obtain acceptable configuration parameters for the DOTS server.

```
      Header: GET (Code=0.01)
      Uri-Host: "host"
      Uri-Path: ".well-known"
      Uri-Path: "dots"
      Uri-Path: "version"
      Uri-Path: "config"
```

Figure ~~15:~~ **17:** GET to retrieve configuration

```
    The DOTS server in the 2.05 (Content) response conveys the current,
    minimum, and maximum attribute values acceptable by the DOTS server
```
(Figure ~~16).~~ **18).**

```
    Content-Format: "application/cbor"
    {
```
      ~~"attack-time-config":~~
      "signal-config": {
        "mitigating-config": {
```
          "heartbeat-interval": {
            "current-value": integer,
            "min-value": integer,
            "max-value": integer
          },
          "missing-hb-allowed": {
            "current-value": integer,
            "min-value": integer,
            "max-value": integer
          },
          "max-retransmit": {
            "current-value": integer,
            "min-value": integer,
            "max-value": integer
          },
          "ack-timeout": {
            "current-value": integer,
            "min-value": integer,
            "max-value": integer
          },
          "ack-random-factor": {
```
            ~~"current-value":~~

```
              "current-value-decimal": number,
              "min-value":
              "min-value-decimal": number,
              "max-value":
              "max-value-decimal": number
            }
          },
          "peace-time-config":
          "idle-config": {
            "heartbeat-interval": {
              "current-value": integer,
              "min-value": integer,
              "max-value": integer
            },
            "missing-hb-allowed": {
              "current-value": integer,
              "min-value": integer,
              "max-value": integer
            },
            "max-retransmit": {
              "current-value": integer,
              "min-value": integer,
              "max-value": integer
            },
            "ack-timeout": {
              "current-value": integer,
              "min-value": integer,
              "max-value": integer
            },
            "ack-random-factor": {
              "current-value":
              "current-value-decimal": number,
              "min-value":
              "min-value-decimal": number,
              "max-value":
              "max-value-decimal": number
            }
          },
          "trigger-mitigation": {
            "current-value": boolean
          },
            "config-interval": {
              "current-value": integer,
              "min-value": integer,
              "max-value": integer
          }
        }
      }
    }
```

                        Figure 16 **18:** GET response body

   Figure 17 **19** shows an example of acceptable and current configuration
   parameters on a DOTS server for DOTS signal channel session
   configuration. The same acceptable configuration is used during
   attack and peace times.

```
   Content-Format: "application/cbor"
   {
        "attack-time-config":
      "signal-config": {
        "mitigating-config": {
          "heartbeat-interval": {
            "current-value": 30,
            "min-value": 15,
            "max-value": 240
          },
          "missing-hb-allowed": {
            "current-value": 5,
            "min-value": 3,
            "max-value": 9
          },
          "max-retransmit": {
            "current-value": 3,
            "min-value": 2,
            "max-value": 15
          },
          "ack-timeout": {
            "current-value": 2,
            "min-value": 1,
            "max-value": 30
          },
          "ack-random-factor": {
            "current-value":
            "current-value-decimal": 1.5,
            "min-value":
            "min-value-decimal": 1.1,
            "max-value":
            "max-value-decimal": 4.0
          }
        },
        "peace-time-config":
        "idle-config": {
          "heartbeat-interval": {
            "current-value": 30,
            "min-value": 15,
            "max-value": 240
          },
          "missing-hb-allowed": {
            "current-value": 5,
            "min-value": 3,
```

```
          "max-value": 9
        },
        "max-retransmit": {
          "current-value": 3,
          "min-value": 2,
          "max-value": 15
        },
        "ack-timeout": {
          "current-value": 2,
          "min-value": 1,
          "max-value": 30

        },
        "ack-random-factor": {
          "current-value":
          "current-value-decimal": 1.5,
          "min-value":
          "min-value-decimal": 1.1,
          "max-value":
          "max-value-decimal": 4.0
        }
      },
      "trigger-mitigation": {
        "current-value": true
      },
        "config-interval": {
          "current-value": 1439,
          "min-value": 0,
          "max-value": 65535
      }
    }
  }
```

                Figure 17: 19: Configuration response body

4.5.2. Convey DOTS Signal Channel Session Configuration

   A PUT request is used to convey the configuration parameters for the
   signal channel (e.g., heartbeat interval, maximum retransmissions).
   Message transmission parameters for CoAP are defined in Section 4.8
   of [RFC7252]. The RECOMMENDED values of transmission parameter
   values are ack-timeout (2 seconds), max-retransmit (3), ack-random-
   factor (1.5). In addition to those parameters, the RECOMMENDED
   specific DOTS transmission parameter values are 'heartbeat-interval'
   (30 seconds) and 'missing-hb-allowed' (5).

      Note: heartbeat-interval should be tweaked to also assist DOTS
      messages for NAT traversal (SIG-010 of
      [I-D.ietf-dots-requirements]). According to [RFC8085], keepalive
      messages must not be sent more frequently than once every 15
      seconds and should use longer intervals when possible.
      Furthermore, [RFC4787] recommends NATs to use a state timeout of 2
      minutes or longer, but experience shows that sending packets every
      15 to 30 seconds is necessary to prevent the majority of
      middleboxes from losing state for UDP flows. From that
      standpoint, this specification recommends a minimum heartbeat-
      interval of 15 seconds and a maximum heartbeat-interval of 240
      seconds. The recommended value of 30 seconds is selected to
      anticipate the expiry of NAT state.

      A heartbeat-interval of 30 seconds may be seen as too chatty in
      some deployments. For such deployments, DOTS agents may negotiate
      longer heartbeat-interval values to prevent any network overload
      with too frequent keepalives.

      **Different heartbeat intervals can be defined for 'mitigation-
      config' and 'idle-config' to reduce being too chatty during idle
      times. If there is an on-path translator between the DOTS client
      (standalone or part of a DOTS gateway) and the DOTS server, the
      'mitigation-config' heartbeat-interval has to be smaller than the
      translator session timeout. It is recommended that the 'idle-
      config' heartbeat-interval is also smaller than the translator
      session timeout to prevent translator transversal issues, or set
      to '0'. Means to discover the lifetime assigned by a translator
      are out of scope.**

   When a confirmable "CoAP Ping" is sent, and if there is no response,
   the "CoAP Ping" is retransmitted max-retransmit number of times by
   the CoAP layer using an initial timeout set to a random duration
   between ack-timeout and (ack-timeout*ack-random-factor) and
   exponential back-off between retransmissions. By choosing the
   recommended transmission parameters, the "CoAP Ping" will timeout
   after 45 seconds. If the DOTS agent does not receive any response
   from the peer DOTS agent for 'missing-hb-allowed' number of
   consecutive "CoAP Ping" confirmable messages, it concludes that the
   DOTS signal channel session is disconnected. A DOTS client MUST NOT
   transmit a "CoAP Ping" while waiting for the previous "CoAP Ping"
   response from the same DOTS server.

   If the DOTS agent wishes to change the default values of message
   transmission parameters, then it should follow the guidance given in
   Section 4.8.1 of [RFC7252]. The DOTS agents MUST use the negotiated
   values for message transmission parameters and default values for
   non-negotiated message transmission parameters.

   The signal channel session configuration is applicable to a single
   DOTS signal channel session between the DOTS agents.

      Header: PUT (Code=0.03)
      Uri-Host: "host"

```
              Uri-Path: ".well-known"
              Uri-Path: "dots"
              Uri-Path: "version"
              Uri-Path: "config"
              Uri-Query: "session-id=123"
              Content-Format: "application/cbor"
              {
                "signal-config": {
                  "session-id": integer,
                  "attack-time-config":
                  "mitigating-config": {
                    "heartbeat-interval": {
                      "current-value": integer
                    },
                    "missing-hb-allowed": {
                      "current-value": integer
                    },
                    "max-retransmit": {
                      "current-value": integer
                    },
                    "ack-timeout": {
                      "current-value": integer
                    },
                    "ack-random-factor": {
                      "current-value":
                      "current-value-decimal": number
                    }
                  },
                  "peace-time-config":
                  "idle-config": {
                    "heartbeat-interval": {
                      "current-value": integer
                    },
                    "missing-hb-allowed": {
                      "current-value": integer
                    },
                    "max-retransmit": {
                      "current-value": integer
                    },
                    "ack-timeout": {
                      "current-value": integer
                    },
                    "ack-random-factor": {
                      "current-value":
                      "current-value-decimal": number
                    }
                  },
                  "trigger-mitigation": boolean,
                  "config-interval": integer
                }
              }
```

              Figure 18: 20: PUT to convey the DOTS signal channel session
                               configuration data.

      The parameters in Figure 18 20 are described below:

      session-id: Identifier for the DOTS signal channel session
         configuration data represented as an integer. This identifier
         MUST be generated by the DOTS client. This document does not make
         any assumption about how this identifier is generated.

         This is a mandatory attribute.

      attack-time-config:

      mitigation-config: Set of configuration parameters to use when an
         attack a
         mitigation is active. The following parameters may be included:

         heartbeat-interval: Time interval in seconds between two
            consecutive heartbeat messages.

            '0' is used to disable the heartbeat mechanism.

            This is an optional attribute.

         missing-hb-allowed: Maximum number of consecutive heartbeat
            messages for which the DOTS agent did not receive a response
            before concluding that the session is disconnected.

            This is an optional attribute.

         max-retransmit: Maximum number of retransmissions for a message
            (referred to as MAX_RETRANSMIT parameter in CoAP).

            This is an optional attribute.

         ack-timeout: Timeout value in seconds used to calculate the
            initial retransmission timeout value (referred to as
            ACK_TIMEOUT parameter in CoAP).

            This is an optional attribute.

         ack-random-factor: Random factor used to influence the timing of
            retransmissions (referred to as ACK_RANDOM_FACTOR parameter in
            CoAP).

            This is an optional attribute.

```
       peace-time-config:

     idle-config:    Set of configuration parameters to use during
         peacetime. when no
         mitigation is active.  This attribute has the same structure as 'attack-time-
         config'.
         'mitigating-config'.

     trigger-mitigation:    If the parameter value is set to 'false', then
         DDoS mitigation is triggered only when the DOTS signal channel
         session is lost.  Automated mitigation on loss of signal is
         discussed in Section 3.3.3 of [I-D.ietf-dots-architecture].

         If the DOTS client ceases to respond to heartbeat messages, the
         DOTS server can detect that the DOTS session is lost.

         The default value of the parameter is 'true'.

         This is an optional attribute.

     config-interval:    This parameter is returned to indicate the time
         interval expressed in minutes, seconds, which a DOTS agent must wait for
         before re-contacting its peer in order to retrieve the signal
         channel configuration data.

         '0' is used to disable this refresh mechanism.

         If a non-null non-zero value of 'config-interval' is received by a DOTS
         agent,
         client, it has to issue a PUT request to refresh the configuration
         parameters for the signal channel before the expiry of 'config-
         interval'.  When a DDoS attack is active, refresh requests MUST
         NOT be sent by DOTS clients and the DOTS server MUST NOT terminate
         the (D)TLS session after the expiry of 'config-interval'.

         This mechanism allows to update the configuration data if a change
         occurs at the DOTS server side.  For example, the new
         configuration may instruct a DOTS client to cease heartbeats or
         reduce heartbeat frequency.

         If this parameter is not returned, this is equivalent to receiving
         a 'config-interval' value set to '0'.

         If a DOTS server detects that a misbehaving DOTS client does not
         contact the DOTS server after the expiry of 'config-interval', in
         order to retrieve the signal channel configuration data, it MAY
         terminate the (D)TLS session.  A (D)TLS session is terminated by
         the receipt of an authenticated message that closes the connection
         (e.g., a fatal alert (Section 7.2 of [RFC5246])).

         This is an optional attribute.

     At least one of the attributes 'heartbeat-interval', 'missing-hb-
     allowed', 'max-retransmit', 'ack-timeout', 'ack-random-factor', and
     'trigger-mitigation' MUST be present in the PUT request.  The PUT
     request with a higher numeric 'session-id' value overrides the DOTS
     signal channel session configuration data installed by a PUT request
     with a lower numeric 'session-id' value.

     Figure 19 21 shows a PUT request example to convey the configuration
     parameters for the DOTS signal channel.  In this example, heartbeat
     mechanism is disabled during peacetime, when no mitigation is active, while the
     heartbeat interval is set to '91' when an attack a mitigation is active.

        Header: PUT (Code=0.03)
        Uri-Host: "www.example.com"
        Uri-Path: ".well-known"
        Uri-Path: "dots"
        Uri-Path: "v1"
        Uri-Path: "config"
        Uri-Query: "session-id=123"
        Content-Format: "application/cbor"
        {
          "signal-config": {
            "session-id": 1234534333242,
            "attack-time-config":
            "mitigating-config": {
               "heartbeat-interval": {
                 "current-value": 91
               },
               "missing-hb-allowed": {
                 "current-value": 3
               },
               "max-retransmit": {
                 "current-value": 7
               },
               "ack-timeout": {
                 "current-value": 5
               },
               "ack-random-factor": {
                 "current-value":
                 "current-value-decimal": 1.5
               }
            },
            "peace-time-config":
            "idle-config": {
               "heartbeat-interval": {
                 "current-value": 0
               },
               "max-retransmit": {
```

```
            "current-value": 7
          },
          "ack-timeout": {
            "current-value": 5
          },
          "ack-random-factor": {
            "current-value":
            "current-value-decimal": 1.5
          }
        },
        "trigger-mitigation": false
      }
    }
```

          Figure 19: 21: PUT to convey the configuration parameters

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

   o If the DOTS server finds the 'session-id' parameter value conveyed
      in the PUT request in its configuration data and if the DOTS
      server has accepted the updated configuration parameters, then
      2.04 (Changed) code is returned in the response.

   o If the DOTS server does not find the 'session-id' 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) is returned in the response.

   o If the request is missing one or more mandatory attributes or it
      contains one or more invalid or unknown parameters, 4.00 (Bad
      Request) is returned in the response.

   o Response code 4.22 (Unprocessable Entity) is returned in the
      response, if any of the 'heartbeat-interval', 'missing-hb-
      allowed', 'max-retransmit', 'target-protocol', 'ack-timeout', and
      'ack-random-factor' attribute values are not acceptable to the
      DOTS server. Upon receipt of the 4.22 error response code, the
      DOTS client should request the maximum and minimum attribute
      values acceptable to the DOTS server (Section 4.5.1).

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

4.5.3. Delete DOTS Signal Channel Session Configuration

   A DELETE request is used to delete the installed DOTS signal channel
   session configuration data (Figure 20). 22).

```
   Header: DELETE (Code=0.04)
   Uri-Host: "host"
   Uri-Path: ".well-known"
   Uri-Path: "dots"
   Uri-Path: "version"
   Uri-Path: "config"
   Content-Format: "application/cbor"
   Uri-Query: "session-id=123"
```

                      Figure 20: 22: DELETE configuration

   The DOTS server resets the DOTS signal channel session configuration
   back to the default values and acknowledges a DOTS client's request
   to remove the DOTS signal channel session configuration using 2.02
   (Deleted) response code.

4.6. Redirected Signaling

   Redirected DOTS signaling is discussed in detail in Section 3.2.2 of
   [I-D.ietf-dots-architecture].

   If a DOTS server wants to redirect a DOTS client to an alternative
   DOTS server for a signal session, then the response code 3.00
   (alternate server) will be returned in the response to the DOTS
   client.

   The DOTS server can return the error response code 3.00 in response
   to a PUT request from the DOTS client or convey the error response
   code 3.00 in a unidirectional notification response from the DOTS
   server.

   The DOTS server in the error response conveys the alternate DOTS
   server's FQDN, and the alternate DOTS server's IP address(es) and
   time to live values in the CBOR body (Figure 21). 23).

```
   {
       "alt-server": "string",
       "alt-server-record": [
         {
           "addr": "string",
           "ttl" : integer
         }
       ]
   }
```

                       Figure 21: 23: Error response body

   The parameters are described below:

   alt-server: FQDN of an alternate DOTS server.

```
    addr:  IP address of an alternate DOTS server.

    ttl:  Time to live (TTL) represented as an integer number of seconds.

    Figure 22 24 shows a 3.00 response example to convey the DOTS alternate
    server 'alt-server.example', its IP addresses 2001:db8:6401::1 and
    2001:db8:6401::2, and TTL values 3600 and 1800.

    {
        "alt-server": "alt-server.example",
        "alt-server-record": [
          {
            "ttl" :  3600,
            "addr": "2001:db8:6401::1"
          },
          {
            "ttl" :  1800,
            "addr": "2001:db8:6401::2"
          }
        ]
    }
```

Figure 22+ **24:** Example of error response body

When the DOTS client receives 3.00 response, it considers the current
request as failed, but SHOULD try re-sending the request to the
alternate DOTS server. During a ~~DDOS~~ **DDoS** attack, the DNS server may be
the target of another DDoS attack, alternate DOTS server's IP
addresses conveyed in the 3.00 response help the DOTS client skip DNS
lookup of the alternate DOTS server. The DOTS client can then try to
establish a UDP or a TCP session with the alternate DOTS server. The
DOTS client SHOULD implement a DNS64 function to handle the scenario
where an IPv6-only DOTS client communicates with an IPv4-only
alternate DOTS server.

4.7. Heartbeat Mechanism

To provide an indication of signal health and distinguish an 'idle'
signal channel from a 'disconnected' or 'defunct' session, the DOTS
agent sends a heartbeat over the signal channel to maintain its half
of the channel. The DOTS agent similarly expects a heartbeat from
its peer DOTS agent, and may consider a session terminated in the
prolonged absence of a peer agent heartbeat.

While the communication between the DOTS agents is quiescent, the
DOTS client will probe the DOTS server to ensure it has maintained
cryptographic state and vice versa. Such probes can also keep
firewalls and/or stateful translators bindings alive. This probing
reduces the frequency of establishing a new handshake when a DOTS
signal needs to be conveyed to the DOTS server.

~~In order to avoid complications due to the presence of some stateful~~
~~translators and firewalls (e.g., discard an incoming packet because~~
~~no matching state is found),~~

DOTS servers MAY trigger their heartbeat requests immediately after
receiving heartbeat probes from peer DOTS clients. **As a reminder, it
is the responsibility of DOTS clients to ensure that on-path
translators/firewalls are maintaining a binding so that the same
external IP address and/or port number is retained for the DOTS
session.**

In case of a massive DDoS attack that saturates the incoming link(s)
to the DOTS client, all traffic from the DOTS server to the DOTS
client will likely be dropped, although the DOTS server receives
heartbeat requests in addition to DOTS messages sent by the DOTS
client. In this scenario, the DOTS agents MUST behave differently to
handle message transmission and DOTS session liveliness during link
saturation:

o The DOTS client MUST NOT consider the DOTS session terminated even
   after a maximum 'missing-hb-allowed' threshold is reached. The
   DOTS client SHOULD keep on using the current DOTS session to send
   heartbeat requests over it, so that the DOTS server knows the DOTS
   client has not disconnected the DOTS session.

   After the maximum 'missing-hb-allowed' threshold is reached, the
   DOTS client SHOULD try to resume the (D)TLS session. The DOTS
   client SHOULD send mitigation requests over the current DOTS
   session, and in parallel, for example, try to resume the (D)TLS
   session or use 0-RTT mode in DTLS 1.3 to piggyback the mitigation
   request in the ClientHello message.

   As soon as the link is no longer saturated, if traffic from the
   DOTS server reaches the DOTS client over the current DOTS session,
   the DOTS client can stop (D)TLS session resumption or if (D)TLS
   session resumption is successful then disconnect the current DOTS
   session.

o If the DOTS server does not receive any traffic from the peer DOTS
   client, then the DOTS server sends heartbeat requests to the DOTS
   client and after maximum 'missing-hb-allowed' threshold is
   reached, the DOTS server concludes the session is disconnected.

In DOTS over UDP, heartbeat messages MUST be exchanged between the
DOTS agents using the "CoAP Ping" mechanism defined in Section 4.2 of
[RFC7252]. Concretely, the DOTS agent sends an Empty Confirmable
message and the peer DOTS agent will respond by sending a Reset
message.

```
        In DOTS over TCP, heartbeat messages MUST be exchanged between the
        DOTS agents using the Ping and Pong messages specified in Section 4.4
        of [I-D.ietf-core-coap-tcp-tls].  That is, the DOTS agent sends a
        Ping message and the peer DOTS agent would respond by sending a
        single Pong message.

    5.  DOTS Signal Channel YANG Module

        This document defines a YANG [RFC7950] module for mitigation scope
        and DOTS signal channel session configuration data.

    5.1.  Tree Structure

        This document defines the YANG module "ietf-dots-signal"
        (Section 5.2), which has the following tree structure.  A DOTS signal
        message can either be a mitigation or a configuration message.

        module: ietf-dots-signal
          +--rw dots-signal
             +--rw (message-type)?
                +--:(mitigation-scope)
                |  +--rw client-identifier*    binary client-domain-hash?    string
                |  +--rw scope* [mitigation-id] [cuid mitigation-id]
                |     +--rw cuid                      string
                |     +--rw mitigation-id        int32
                |     +--rw target-prefix*       inet:ip-prefix
                |     +--rw target-port-range* [lower-port upper-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 lifetime?            int32
                |     +--rw mitigation-start?    int64
                |     +--ro status?              enumeration
                |     +--ro conflict-information
                |     |  +--ro conflict-status?   enumeration
                |     |  +--ro conflict-cause?    enumeration
                |     |  +--ro retry-timer?       int32
                |     |  +--ro conflict-scope
                |     |     +--ro target-prefix*       inet:ip-prefix
                |     |     +--ro target-port-range* [lower-port upper-port]
                |     |     |  +--ro lower-port    inet:port-number
                |     |     |  +--ro upper-port    inet:port-number
                |     |     +--ro target-protocol*     uint8
                |     |     +--ro target-fqdn*         inet:domain-name
                |     |     +--ro target-uri*          inet:uri
                |     |     +--ro alias-name*          string
                |     |     +--ro acl-list* [acl-name acl-type]
                |     |        +--ro acl-name    -> /ietf-acl:access-lists/acl/acl-name
                |     |        +--ro acl-type    -> /ietf-acl:access-lists/acl/acl-type
                |     +--ro pkts-dropped?        yang:zero-based-counter64
                |     +--ro bps-dropped?         yang:zero-based-counter64
                |     +--ro bytes-dropped?       yang:zero-based-counter64
                |     +--ro pps-dropped?         yang:zero-based-counter64
                +--:(configuration)
                |     +--rw attack-status?         enumeration
                +--:(signal-config)
                |  +--rw session-id         int32
                |  +--rw attack-time-config mitigating-config
                |  |  +--rw heartbeat-interval
                |  |  |  +--rw max-value?      int16
                |  |  |  +--rw min-value?      int16
                |  |  |  +--rw current-value?  int16
                |  |  +--rw missing-hb-allowed
                |  |  |  +--rw max-value?      int16
                |  |  |  +--rw min-value?      int16
                |  |  |  +--rw current-value?  int16
                |  |  +--rw max-retransmit
                |  |  |  +--rw max-value?      int16
                |  |  |  +--rw min-value?      int16
                |  |  |  +--rw current-value?  int16
                |  |  +--rw ack-timeout
                |  |  |  +--rw max-value?      int16
                |  |  |  +--rw min-value?      int16
                |  |  |  +--rw current-value?  int16
                |  |  +--rw ack-random-factor
                |  |     +--rw max-value? max-value-decimal?      decimal64
                |  |     +--rw min-value? min-value-decimal?      decimal64
                |  |     +--rw current-value? current-value-decimal?   decimal64
                |  +--rw peace-time-config idle-config
                |  |  +--rw heartbeat-interval
                |  |  |  +--rw max-value?      int16
                |  |  |  +--rw min-value?      int16
                |  |  |  +--rw current-value?  int16
                |  |  +--rw missing-hb-allowed
                |  |  |  +--rw max-value?      int16
                |  |  |  +--rw min-value?      int16
                |  |  |  +--rw current-value?  int16
                |  |  +--rw max-retransmit
                |  |  |  +--rw max-value?      int16
                |  |  |  +--rw min-value?      int16
                |  |  |  +--rw current-value?  int16
                |  |  +--rw ack-timeout
                |  |  |  +--rw max-value?      int16
                |  |  |  +--rw min-value?      int16
                |  |  |  +--rw current-value?  int16
                |  |  +--rw ack-random-factor
                |  |     +--rw max-value? max-value-decimal?      decimal64
```

```
            | |      +--rw min-value? min-value-decimal?       decimal64
            | |      +--rw current-value? current-value-decimal?  decimal64
            | +--rw trigger-mitigation?  boolean
            | +--rw config-interval?    int32
            +--:(redirected-signal)
               +--rw alt-server          string
               +--rw alt-server-record* [addr]
                  +--rw addr    inet:ip-address
                  +--rw ttl?   int32
```

5.2. YANG Module

 <CODE BEGINS> file "ietf-dots-signal@2017-12-19.yang" "ietf-dots-signal@2018-01-09.yang"

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

   import ietf-inet-types {prefix "inet";}
   import ietf-yang-types {prefix yang;}
   import ietf-access-control-list {prefix "ietf-acl";}

   organization "IETF DDoS Open Threat Signaling (DOTS) Working Group";

   contact
     "Konda, Tirumaleswar Reddy <TirumaleswarReddy_Konda@McAfee.com>
      Mohamed Boucadair <mohamed.boucadair@orange.com>
      Prashanth Patil <praspati@cisco.com>
      Andrew Mortensen <amortensen@arbor.net>
      Nik Teague <nteague@verisign.com>";

   description
     "This module contains YANG definition for the signaling
      messages exchanged between a DOTS client and a DOTS server.

      Copyright (c) 2017 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 2017-12-19 2018-01-09 {
     description
       "Initial revision.";
     reference
       "RFC XXXX: Distributed Denial-of-Service Open Threat
                  Signaling (DOTS) Signal Channel";
   }

   grouping target {
     description
       "Specifies the scope targets of the mitigation request.";

     leaf-list target-prefix {
       type inet:ip-prefix;
       description
        "IPv4 or IPv6 prefix identifying the target.";
     }

     list target-port-range {
       key "lower-port upper-port";

       description
         "Port range. When only lower-port is
          present, it represents a single port.";

       leaf lower-port {
         type inet:port-number;
         mandatory true;
         description "Lower port number.";
       }

       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.";
       }
     }

     leaf-list target-protocol {
       type uint8;
       description
         "Identifies the target protocol number.

          The value '0' means 'all protocols'.

          Values are taken from the IANA protocol registry:
```

```
              https://www.iana.org/assignments/protocol-numbers/
              protocol-numbers.xhtml

              For example, 6 for TCP or 17 for UDP.";
        }

        leaf-list target-fqdn {
          type inet:domain-name;
          description "FQDN identifying the target.";
        }

        leaf-list target-uri {
          type inet:uri;
          description "URI identifying the target.";
        }

        leaf-list alias-name {
          type string;
          description "alias name";
        }
      }

      grouping mitigation-scope {
        description
          "Specifies the scope of the mitigation request.";

          leaf-list client-identifier

        leaf client-domain-hash {
          type binary; string;
          description
            "The client identifier domain hash may be conveyed by
             the server-domain DOTS gateway to propagate the DOTS
             client domain identification information from the
             gateway's client-side to the gateway's server-side,
             and from the gateway's server-side to the DOTS
             server.

             It may be used by the final DOTS server
             for policy enforcement purposes.";
        }

        list scope {
          key mitigation-id; "cuid mitigation-id";
          description
            "The scope of the request.";

          leaf cuid {
            type string;
            description
              "A unique identifier that is randomly
               generated by a DOTS client to prevent
               request collisions.";
          }

          leaf mitigation-id {
            type int32;
            description
              "Mitigation request identifier.

               This identifier must be unique for each mitigation
               request bound to the DOTS client.";
          }

          uses target;

          leaf lifetime {
            type int32;
            units "seconds";
            default 3600;
            description
              "Indicates the lifetime of the mitigation request.";
            reference
              "RFC XXXX: Distributed Denial-of-Service Open Threat
                         Signaling (DOTS) Signal Channel";
          }

          leaf mitigation-start {
            type int64;
            units "seconds";
            description
              "Mitigation start time is represented in seconds
               relative to 1970-01-01T00:00Z in UTC time.";
          }

          leaf status {
            type enumeration {
              enum "attack-mitigation-in-progress"  {
                value 1;
                description
                  "Attack mitigation is in progress (e.g., changing
                   the network path to re-route the inbound traffic
                   to DOTS mitigator).";
              }

              enum "attack-successfully-mitigated" {
                value 2;
                description
                  "Attack is successfully mitigated (e.g., traffic
                   is redirected to a DDOS DDoS mitigator and attack
```

```
                traffic is dropped or blackholed).";
          }
          enum "attack-stopped" {
            value 3;
            description
              "Attack has stopped and the DOTS client can
               withdraw the mitigation request.";
          }

          enum "attack-exceeded-capability" {
            value 4;
            description
              "Attack has exceeded the mitigation provider
               capability.";
          }

          enum "dots-client-withdrawn-mitigation" {
            value 5;
            description
              "DOTS client has withdrawn the mitigation
               request and the mitigation is active but
               terminating.";
          }

          enum "attack-mitigation-terminated" {
            value 6;
            description
              "Attack mitigation is now terminated.";
          }

          enum "attack-mitigation-withdrawn" {
            value 7;
            description
              "Attack mitigation is withdrawn.";
          }

          enum "attack-mitigation-rejected" {
            value 8;
            description
              "Attack mitigation is rejected.";
          }
        }
      config false;
      description
        "Indicates the status of a mitigation request.

         It must be included in responses only.";
      }

      container conflict-information {
        config false;
        description
          "Indicates that a conflict is detected.
           Must only be used for responses.";

        leaf conflict-status {
          type enumeration {
            enum "request-inactive-other-active"  {
              value 1;
              description
                "DOTS Server has detected conflicting mitigation
                 requests from different DOTS clients.
                 This mitigation request is currently inactive
                 until the conflicts are resolved. Another
                 mitigation request is active.";
            }

            enum "request-active" {
              value 2;
              description
                "DOTS Server has detected conflicting mitigation
                 requests from different DOTS clients.
                 This mitigation request is currently active.";
            }

            enum "all-requests-inactive" {
              value 3;
              description
                "DOTS Server has detected conflicting mitigation
                 requests from different DOTS clients.  All
                 conflicting mitigation requests are inactive.";
            }
          }
          description
            "Indicates the conflict status.
             It must be included in responses only.";
        }

        leaf conflict-cause {
          type enumeration {
            enum "overlapping-targets"  {
              value 1;
              description
                "Overlapping targets. conflict-scope provides
                 more details about the exact conflict.";
            }

            enum "conflict-with-whitelist" {
              value 2;
```

```
                  description
                    "Conflicts with an existing white list.

                     This code is returned when the DDoS mitigation
                     detects that some of the source addresses/prefixes
                     listed in the white list ACLs are actually
                     attacking the target.";
              }
          }
          description
            "Indicates the cause of the conflict.
             It must be included in responses only.";
      }

      leaf retry-timer {
        type int32;
        units "seconds";
        description
          "The DOTS client must not re-send the
           same request before the expiry of this timer.
           It must be included in responses, only.";
      }

      container conflict-scope {
        description
          "Provides more information about the conflict scope.";

        uses target {
          when "../conflict-cause = 'overlapping-targets'";
        }

        list acl-list {
          when "../../conflict-cause = 'conflict-with-whitelist'";
          key "acl-name acl-type";
          description
            "List of conflicting ACLs";

          leaf acl-name {
            type leafref {
              path "/ietf-acl:access-lists/ietf-acl:acl" +
                   "/ietf-acl:acl-name";
            }
            description
              "Reference to the conflicting ACL name bound to
               a DOTS client.";
          }

          leaf acl-type {
            type leafref {
              path "/ietf-acl:access-lists/ietf-acl:acl" +
                   "/ietf-acl:acl-type";
            }
            description
              "Reference to the conflicting ACL type bound to
               a DOTS client.";
          }
        }
      }
    }

    leaf pkts-dropped {
      type yang:zero-based-counter64;
      config false;
      description
        "Number of dropped packets";
    }

    leaf bps-dropped {
      type yang:zero-based-counter64;
      config false;
      description
        "The average number of dropped bytes per second for
         the mitigation request since the attack
         mitigation is triggered.";
    }

    leaf bytes-dropped {
      type yang:zero-based-counter64;
      units 'bytes';
      config false;
      description
        "Counter for dropped packets; in bytes.";
    }

    leaf pps-dropped {
      type yang:zero-based-counter64;
      config false;
      description
        "The average number of dropped packets per second
         for the mitigation request since the attack
         mitigation is triggered.";
    }
}

leaf attack-status {
  type enumeration {
    enum "under-attack"  {
      value 1;
      description
        "The DOTS client determines that it is still under
```

```
                    attack.";
               }

              enum "attack-successfully-mitigated" {
                value 2;
                 description
                   "The DOTS client determines that the attack is
                    successfully mitigated.";
              }
            }
            description
              "Indicates the status of an attack as seen by the
               DOTS client.";
          }
      }
    }

    grouping config-parameters {
      description
        "Subset of DOTS signal channel session configuration.";

      container heartbeat-interval {
        description
          "DOTS agents regularly send heartbeats to each other
           after mutual authentication is successfully
           completed in order to keep the DOTS signal channel
           open.";

        leaf max-value {
          type int16;
          units "seconds";
          description
            "Maximum acceptable value.";
          reference
            "RFC XXXX: Distributed Denial-of-Service Open Threat
                       Signaling (DOTS) Signal Channel";
        }

        leaf min-value {
          type int16;
          units "seconds";
          description
            "Minimum acceptable value.";
          reference
            "RFC XXXX: Distributed Denial-of-Service Open Threat
                       Signaling (DOTS) Signal Channel";
        }
        leaf current-value {
          type int16;
          units "seconds";
          default 30;
          description
            "Current value.

             '0' means that heartbeat mechanism is deactivated.";
          reference
            "RFC XXXX: Distributed Denial-of-Service Open Threat
                       Signaling (DOTS) Signal Channel";
        }
      }

      container missing-hb-allowed {
        description
          "Maximum number of missing heartbeats allowed.";

        leaf max-value {
          type int16;
          description
            "Maximum acceptable value.";
          reference
            "RFC XXXX: Distributed Denial-of-Service Open Threat
                       Signaling (DOTS) Signal Channel";
        }

        leaf min-value {
          type int16;
          description
            "Minimum acceptable value.";
          reference
            "RFC XXXX: Distributed Denial-of-Service Open Threat
                       Signaling (DOTS) Signal Channel";
        }
        leaf current-value {
          type int16;
          default 5;
          description
            "Current value.";
          reference
            "RFC XXXX: Distributed Denial-of-Service Open Threat
                       Signaling (DOTS) Signal Channel";
        }
      }

      container max-retransmit {
        description
          "Maximum number of retransmissions of a Confirmable
           message.";

        leaf max-value {
          type int16;
```

```
        description
          "Maximum acceptable value.";
        reference
          "Section 4.8 of RFC 7552.";
      }

      leaf min-value {
        type int16;
        description
          "Minimum acceptable value.";
        reference
          "Section 4.8 of RFC 7552.";
      }
      leaf current-value {
        type int16;
        default 3;
        description
          "Current value.";
        reference
          "RFC XXXX: Distributed Denial-of-Service Open Threat
                    Signaling (DOTS) Signal Channel";
      }
    }

    container ack-timeout {
      description
        "Initial retransmission timeout value.";

      leaf max-value {
        type int16;
        units "seconds";
        description
          "Maximum value.";
        reference
          "Section 4.8 of RFC 7552.";
      }
      leaf min-value {
        type int16;
        units "seconds";
        description
          "Minimum value.";
        reference
          "Section 4.8 of RFC 7552.";
      }
      leaf current-value {
        type int16;
        units "seconds";
        default 2;
        description
          "Current value.";
        reference
          "Section 4.8 of RFC 7552.";
      }
    }

    container ack-random-factor {
      description
        "Random factor used to influence the timing of
         retransmissions.";

      leaf max-value max-value-decimal {
        type decimal64 {
          fraction-digits 2;
        }
        description
          "Maximum acceptable value.";
        reference
          "Section 4.8 of RFC 7552.";
      }

      leaf min-value min-value-decimal {
        type decimal64 {
          fraction-digits 2;
        }
        description
          "Minimum acceptable value.";
        reference
          "Section 4.8 of RFC 7552.";
      }
      leaf current-value current-value-decimal {
        type decimal64 {
          fraction-digits 2;
        }
        default 1.5;
        description
          "Current value.";
        reference
          "Section 4.8 of RFC 7552.";
      }
    }
  }

  grouping signal-config {
    description
      "DOTS signal channel session configuration.";

    leaf session-id {
      type int32;
      mandatory true;
      description
```

```
           "An identifier for the DOTS signal channel
            session configuration data.";
     }

     container attack-time-config mitigating-config {
       description
         "Configuration paramaters parameters to use when an attack a mitigation is active.";
       uses config-parameters;
     }

     container peace-time-config idle-config {
       description
         "Configuration paramaters parameters to use in peacetime."; when no mitigation is
          active.";
       uses config-parameters;
     }

     leaf trigger-mitigation {
       type boolean;
       default true;
       description
         "If false, then mitigation is triggered
          only when the DOTS server channel session is lost";
       reference
         "RFC XXXX: Distributed Denial-of-Service Open Threat
                    Signaling (DOTS) Signal Channel";
     }

     leaf config-interval {
       type int32;
       units "minutes"; "seconds";
       description
         "This parameter is returned by a DOTS server to
          a requesting DOTS client to indicate the time interval
          after which the DOTS client must contact the DOTS
          server in order to retrieve the signal channel
          configuration data.

          This mechanism allows the update of the configuration
          data if a change occurs.

          For example, the new configuration may instruct
          a DOTS client to cease heartbeats or reduce
          heartbeat frequency.

          '0' is used to disable this refresh mechanism.";
     }
   }

   grouping redirected-signal {
     description
       "Grouping for the redirected signaling.";

     leaf alt-server {
       type string;
       mandatory true;
       description
         "Alias of an alternate server.";
     }

     list alt-server-record {
       key "addr";
       description
         "List of records for the alternate server.";

       leaf addr {
         type inet:ip-address;
         description
          "IPv4 or IPv6 address identifying the server.";
       }

       leaf ttl {
         type int32;
         description
          "TTL associated with this record.";
       }
     }
   }

   container dots-signal {
     description
       "Main container for DOTS signal message.
        A DOTS signal message can be a mitigation message or
        a configuration message.";

     choice message-type {
       description
           "Either
         "Can be a mitigation mitigation, a configuration, or a configuration redirect
          message.";

       case mitigation-scope {
         description
           "Mitigation scope of a mitigation message.";
         uses mitigation-scope;
       }

       case configuration signal-config {
         description
           "Configuration message.";
```

```
            uses signal-config;
        }

        case redirected-signal {
          description
            "Redirected signaling.";
          uses redirected-signal;
        }
      }
    }
  }
 <CODE ENDS>
```

6. Mapping Parameters to CBOR

   All parameters in the payload of the DOTS signal channel MUST be
   mapped to CBOR types as shown in Table 4 and are assigned an integer
   key to save space. The recipient of the payload MAY reject the
   information if it is not suitably mapped.

```
   /---------------------+---------------+-------------------------\
   | Parameter name      | CBOR key      | CBOR major type of value |
   +---------------------+---------------+-------------------------+
   | mitigation-scope    | 1             | 5 (map)                  |
   | scope               | 2             | 5 (map)                  |
   | mitigation-id       | 3             | 0 (unsigned)             |
   | acl-list            | 4             | 4                        |
   | target-port-range   | 5             | 4                        |
   | lower-port          | 6             | 0                        |
   | upper-port          | 7             | 0                        |
   | target-protocol     | 8             | 4                        |
   | target-fqdn         | 9             | 4                        |
   | target-uri          | 10            | 4                        |
   | alias-name          | 11            | 4                        |
   | lifetime            | 12            | 0                        |
   | attack-status       | 13            | 0                        |
   | signal-config       | 14            | 5                        |
   | heartbeat-interval  | 15            | 5 (map)                  |
   | max-retransmit      | 16            | 5 (map)                  |
   | ack-timeout         | 17            | 5 (map)                  |
   | ack-random-factor   | 18            | 5 (map)                  |
   | min-value           | 19            | 0                        |
   | max-value           | 20            | 0                        |
   | status              | 21            | 0                        |
   | conflict-information| 22            | 5 (map)                  |
   | conflict-status     | 23            | 0                        |
   | conflict-cause      | 24            | 0                        |
   | retry-timer         | 25            | 0                        |
   | bytes-dropped       | 26            | 0                        |
   | bps-dropped         | 27            | 0                        |
   | pkts-dropped        | 28            | 0                        |
   | pps-dropped         | 29            | 0                        |
   | session-id          | 30            | 0                        |
   | trigger-mitigation  | 31            | 7 (simple types)         |
   | missing-hb-allowed  | 32            | 5 (map)                  |
   | current-value       | 33            | 0                        |
   | mitigation-start    | 34            | 7 (floating-point)       |
   | target-prefix       | 35            | 4 (array)                |
   | client-identifier client-domain-hash | 36 | 2 (byte string) 3      |
   | alt-server          | 37            | 2                        |
   | alt-server-record   | 38            | 4                        |
   | addr                | 39            | 2 3                      |
   | ttl                 | 40            | 0                        |
   | conflict-scope      | 41            | 5 (map)                  |
   | acl-name            | 42            | 3                        |
   | acl-type            | 43            | 3                        |
   | config-interval     | 44            | 0                        |
   | attack-time-config mitigating-config | 45 | 5 (map)               |
   | peace-time-config idle-config | 46    | 5 (map)                  |
   | cuid                | 47            | 3                        |
   | min-value-decimal   | 48            | 7                        |
   | max-value-decimal   | 49            | 7                        |
   | current-value-decimal| 50           | 7                        |
   \---------------------+---------------+-------------------------/
      Table 4: CBOR mappings used in DOTS signal channel message
```

7. (D)TLS Protocol Profile and Performance Considerations

7.1. (D)TLS Protocol Profile

   This section defines the (D)TLS protocol profile of DOTS signal
   channel over (D)TLS and DOTS data channel over TLS.

   There are known attacks on (D)TLS, such as man-in-the-middle and
   protocol downgrade attacks. These are general attacks on (D)TLS and,
   as such, they are not specific to DOTS over (D)TLS; refer to the
   (D)TLS RFCs for discussion of these security issues. DOTS agents
   MUST adhere to the (D)TLS implementation recommendations and security
   considerations of [RFC7525] except with respect to (D)TLS version.
   Since DOTS signal channel encryption relies upon (D)TLS is virtually
   a green-field deployment, DOTS agents MUST implement only (D)TLS 1.2
   or later.

   When a DOTS client is configured with a domain name of the DOTS
   server, and connects to its configured DOTS server, the server may
   present it with a PKIX certificate. In order to ensure proper
   authentication, a DOTS client MUST verify the entire certification
   path per [RFC5280]. The DOTS client additionally uses [RFC6125]
   validation techniques to compare the domain name with the certificate
   provided.

A key challenge to deploying DOTS is the provisioning of DOTS
clients, including the distribution of keying material to DOTS
clients to enable the required mutual authentication of DOTS agents.
EST defines a method of certificate enrollment by which domains
operating DOTS servers may provide DOTS clients with all the
necessary cryptographic keying material, including a private key and
a certificate to authenticate themselves. One deployment option is
DOTS clients behave as EST clients for certificate enrollment from an
EST server provisioned by the ~~mitigation provider. This document~~
~~does not specify which EST mechanism~~ **mitigation provider. This document**
does not specify which EST mechanism the DOTS client uses to achieve
initial enrollment.

The Server Name Indication (SNI) extension [RFC6066] defines a
mechanism for a client to tell a (D)TLS server the name of the server
it wants to contact. This is a useful extension for hosting
environments where multiple virtual servers are reachable over a
single IP address. The DOTS client may or may not know if it is
interacting with a DOTS server in the hosting environment, so the
DOTS client ~~uses to achieve~~
~~initial enrollment.~~ **SHOULD include the DOTS server FQDN in the SNI extension.**

Implementations compliant with this profile MUST implement all of the
following items:

o DTLS record replay detection (Section 3.3 of [RFC6347]) to protect
   against replay attacks.

o (D)TLS session resumption without server-side state [RFC5077] to
   resume session and convey the DOTS signal.

o Raw public keys [RFC7250] or PSK handshake [RFC4279] which reduces
   the size of the ServerHello, and can be used by DOTS agents that
   cannot obtain certificates.

Implementations compliant with this profile SHOULD implement all of
the following items to reduce the delay required to deliver a DOTS
signal channel message:

o TLS False Start [RFC7918] which reduces round-trips by allowing
   the TLS second flight of messages (ChangeCipherSpec) to also
   contain the DOTS signal.

o Cached Information Extension [RFC7924] which avoids transmitting
   the server's certificate and certificate chain if the client has
   cached that information from a previous TLS handshake.

o TCP Fast Open [RFC7413] can reduce the number of round-trips to
   convey DOTS signal channel message.

7.2. (D)TLS 1.3 Considerations

TLS 1.3 [I-D.ietf-tls-tls13] provides critical latency improvements
for connection establishment over TLS 1.2. The DTLS 1.3 protocol
[I-D.ietf-tls-dtls13] is based upon the TLS 1.3 protocol and provides
equivalent security guarantees. (D)TLS 1.3 provides two basic
handshake modes the DOTS signal channel can take advantage of:

o A full handshake mode in which a DOTS client can send a DOTS
   mitigation request message after one round trip and the DOTS
   server immediately responds with a DOTS mitigation response. This
   assumes no packet loss is experienced.

o 0-RTT mode in which the DOTS client can authenticate itself and
   send DOTS mitigation request messages in the first message, thus
   reducing handshake latency. 0-RTT only works if the DOTS client
   has previously communicated with that DOTS server, which is very
   likely with the DOTS signal channel.

   The DOTS client has to establish a (D)TLS session with the DOTS
   server during peacetime and share a PSK.

   During a DDoS attack, the DOTS client can use the (D)TLS session
   to convey the DOTS mitigation request message and, if there is no
   response from the server after multiple retries, the DOTS client
   can resume the (D)TLS session in 0-RTT mode using PSK.

   Section 8 of [I-D.ietf-tls-tls13] discusses some mechanisms to
   implement to limit the impact of replay attacks on 0-RTT data. If
   TLS1.3 is used, DOTS servers must implement one of these
   mechanisms.

   A simplified TLS 1.3 handshake with 0-RTT DOTS mitigation request
   message exchange is shown in Figure ~~23.~~ **25.**

         DOTS Client   DOTS Server

      ClientHello
      (Finished)
      (0-RTT DOTS signal message)
      (end_of_early_data)   -------->
                                                       ServerHello
                                               {EncryptedExtensions}
                                               {ServerConfiguration}
                                                        {Certificate}
                                                  {CertificateVerify}
                                                          {Finished}
                                  <--------   [DOTS signal message]
      {Finished}   -------->

```
        [DOTS signal message]     <------->   [DOTS signal message]

                   Figure 23: 25: TLS 1.3 handshake with 0-RTT
```

7.3. MTU and Fragmentation

   To avoid DOTS signal message fragmentation and the subsequent
   decreased probability of message delivery, DOTS agents MUST ensure
   that the DTLS record MUST fit within a single datagram. If the path
   MTU is not known to the DOTS server, an IP MTU of 1280 bytes SHOULD
   be assumed. The length of the URL MUST NOT exceed 256 bytes. If UDP
   is used to convey the DOTS signal messages then the DOTS client must
   consider the amount of record expansion expected by the DTLS
   processing when calculating the size of CoAP message that fits within
   the path MTU. Path MTU MUST be greater than or equal to [CoAP
   message size + DTLS overhead of 13 octets + authentication overhead
   of the negotiated DTLS cipher suite + block padding (Section 4.1.1.1
   of [RFC6347]). If the request size exceeds the path MTU then the
   DOTS client MUST split the DOTS signal into separate messages, for
   example the list of addresses in the 'target-prefix' parameter could
   be split into multiple lists and each list conveyed in a new PUT
   request.

   Implementation Note: DOTS choice of message size parameters works
   well with IPv6 and with most of today's IPv4 paths. However, with
   IPv4, it is harder to reliably ensure that there is no IP
   fragmentation. If IPv4 path MTU is unknown, implementations may want
   to limit themselves to more conservative IPv4 datagram sizes such as
   576 bytes, as per [RFC0791]. IP packets whose size does not exceed
   576 bytes should never need to be fragmented: therefore, sending a
   maximum of 500 bytes of DOTS signal over a UDP datagram will
   generally avoid IP fragmentation.

8. Mutual Authentication of DOTS Agents & Authorization of DOTS Clients

   (D)TLS based upon client certificate can be used for mutual
   authentication between DOTS agents. If a DOTS gateway is involved,
   DOTS clients and DOTS gateways MUST perform mutual authentication;
   only authorized DOTS clients are allowed to send DOTS signals to a
   DOTS gateway. The DOTS gateway and the DOTS server MUST perform
   mutual authentication; a DOTS server only allows DOTS signal channel
   messages from an authorized DOTS gateway, thereby creating a two-link
   chain of transitive authentication between the DOTS client and the
   DOTS server.

   **The DOTS server SHOULD support certificate-based client
   authentication. The DOTS client SHOULD respond to the DOTS server's
   TLS certificate request message with the PKIX certificate held by the
   DOTS client. DOTS client certificate validation MUST be performed as
   per [RFC5280] and the DOTS client certificate MUST conform to the
   [RFC5280] certificate profile. If a DOTS client does not support TLS
   client certificate authentication, it MUST support pre-shared key
   based or raw public key based client authentication.**

```
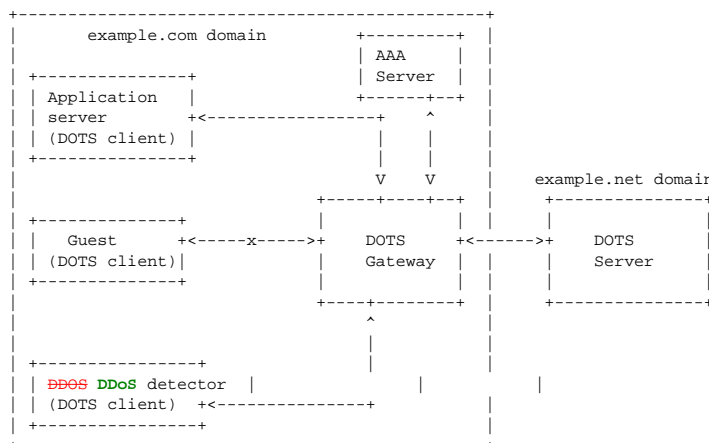   +----------------------------------------------+
   |      example.com domain       +---------+    |
   |                               | AAA     |    |
   | +---------------+             | Server  |    |
   | | Application   |             +------+--+    |
   | | server        +<----------------+   ^      |
   | | (DOTS client) |             |   |   |      |
   | +--------------+              |   |   |      |
   |                               V   V   |  example.net domain
   |                           +-----+----+--+ |  +---------------+
   | +--------------+          |          | | |  |               |
   | |  Guest       +<-----x----->+  DOTS  +<------>+  DOTS       |
   | | (DOTS client)|           | Gateway | | |  |  Server       |
   | +--------------+           |         | | |  |               |
   |                            +----+-------+ |  +---------------+
   |                                 ^       |
   |                                 |       |
   | +----------------+              |       |
   | | DDOS DDoS detector |          |       |
   | | (DOTS client)  +<---------------+     |
   | +----------------+                      |
   +----------------------------------------------+
```

   Figure 24: 26: Example of Authentication and Authorization of DOTS Agents

   In the example depicted in Figure 24, 26, the DOTS gateway and DOTS
   clients within the 'example.com' domain mutually authenticate with
   each other. After the DOTS gateway validates the identity of a DOTS
   client, it communicates with the AAA server in the 'example.com'
   domain to determine if the DOTS client is authorized to request DDoS
   mitigation. If the DOTS client is not authorized, a 4.01
   (Unauthorized) is returned in the response to the DOTS client. In
   this example, the DOTS gateway only allows the application server and
   DDoS attack detector to request DDOS DDoS mitigation, but does not permit
   the user of type 'guest' to request DDoS mitigation.

   Also, DOTS gateways and servers located in different domains MUST
   perform mutual authentication (e.g., using certificates). A DOTS
   server will only allow a DOTS gateway with a certificate for a
   particular domain to request mitigation for that domain. In
   reference to Figure 24, 26, the DOTS server only allows the DOTS gateway
   to request mitigation for 'example.com' domain and not for other
   domains.

9. IANA Considerations

This specification registers a service port (Section 9.1), an URI
suffix in the Well-Known URIs registry (Section 9.2), a CoAP response
code (Section 9.3), a YANG module (Section 9.5). It also creates a
registry for mappings to CBOR (Section 9.4).

9.1. DOTS Signal Channel UDP and TCP Port Number

   IANA is requested to assign the port number TBD to the DOTS signal
   channel protocol for both UDP and TCP from the "Service Name and
   Transport Protocol Port Number Registry" available at
   https://www.iana.org/assignments/service-names-port-numbers/service-
   names-port-numbers.xhtml.

   The assignment of port number 4646 is strongly suggested, as 4646 is
   the ASCII decimal value for ".." (DOTS).

9.2. Well-Known 'dots' URI

   This document requests IANA to register the 'dots' well-known URI in
   the Well-Known URIs registry (https://www.iana.org/assignments/well-
   known-uris/well-known-uris.xhtml) as defined by [RFC5785].

   URI suffix: dots

   Change controller: IETF

   Specification document(s): This RFC

   Related information: None

9.3. CoAP Response ~~Code~~ **Codes**

   IANA is requested to add the following ~~entry~~ **entries** to the "CoAP Response
   Codes" sub-registry available at https://www.iana.org/assignments/
   core-parameters/core-parameters.xhtml#response-codes:

```
                   +------+------------------+-----------+

              +------+------------------+-----------+
              | Code | Description      | Reference |
              +------+------------------+-----------+
              +------+------------------+-----------+
              | 3.00 | Alternate server Server  | [RFCXXXX] |
              | 4.16 | CUID Collision   | [RFCXXXX] |
              | 5.06 | Hop Limit Reached | [RFCXXXX] |
              +------+------------------+-----------+
              +------+------------------+-----------+
```

                     Table 4: CoAP Response Code

9.4. DOTS Signal Channel CBOR Mappings Registry

   The document requests IANA to create a new registry, entitled "DOTS
   Signal Channel CBOR Mappings Registry". The structure of this
   registry is provided in Section 9.4.1.

   The registry is initially populated with the values in Section 9.4.2.

   Values from that registry MUST be assigned via Expert Review
   [RFC8126].

9.4.1. Registration Template

   Parameter name:
      Parameter name as used in the DOTS signal channel.

   CBOR Key Value:
      Key value for the parameter. The key value MUST be an integer in
      the 1-65536 range. The key values in the 32758-65536 range are
      assigned to Vendor-Specific parameters.

   CBOR Major Type:
      CBOR Major type and optional tag for the claim.

   Change Controller:
      For Standards Track RFCs, list the "IESG". For others, give the
      name of the responsible party. Other details (e.g., postal
      address, email address, home page URI) may also be included.

   Specification Document(s):
      Reference to the document or documents that specify the parameter,
      preferably including URIs that can be used to retrieve copies of
      the documents. An indication of the relevant sections may also be
      included but is not required.

9.4.2. Initial Registry Contents

   o Parameter Name: mitigation-scope
   o CBOR Key Value: 1
   o CBOR Major Type: 5
   o Change Controller: IESG
   o Specification Document(s): this document

   o Parameter Name: scope
   o CBOR Key Value: 2
   o CBOR Major Type: 5
   o Change Controller: IESG
   o Specification Document(s): this document
   o Parameter Name: mitigation-id

```
o  CBOR Key Value: 3
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: acl-list
o  CBOR Key Value: 4
o  CBOR Major Type: 4
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: target-port-range
o  CBOR Key Value: 5
o  CBOR Major Type: 4
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: lower-port
o  CBOR Key Value: 6
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: upper-port
o  CBOR Key Value: 7
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: target-protocol
o  CBOR Key Value: 8
o  CBOR Major Type: 4
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: target-fqdn
o  CBOR Key Value: 9
o  CBOR Major Type: 4
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: target-uri
o  CBOR Key Value: 10
o  CBOR Major Type: 4
o  Change Controller: IESG
o  Specification Document(s): this document
o  Parameter Name: alias-name
o  CBOR Key Value: 11
o  CBOR Major Type: 4
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: lifetime
o  CBOR Key Value: 12
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: attack-status
o  CBOR Key Value: 13
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: signal-config
o  CBOR Key Value: 14
o  CBOR Major Type: 5
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: heartbeat-interval
o  CBOR Key Value: 15
o  CBOR Major Type: 5
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: max-retransmit
o  CBOR Key Value: 16
o  CBOR Major Type: 5
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: ack-timeout
o  CBOR Key Value: 17
o  CBOR Major Type: 5
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: ack-random-factor
o  CBOR Key Value: 18
o  CBOR Major Type: 5
o  Change Controller: IESG
o  Specification Document(s): this document
o  Parameter Name: min-value
o  CBOR Key Value: 19
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document

o  Parameter Name: max-value
```

```
o  CBOR Key Value: 20
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: status
o  CBOR Key Value: 21
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: conflict-information
o  CBOR Key Value: 22
o  CBOR Major Type: 5
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: conflict-status
o  CBOR Key Value: 23
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: conflict-cause
o  CBOR Key Value: 24
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: retry-timer
o  CBOR Key Value: 25
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: bytes-dropped
o  CBOR Key Value: 26
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document
o  Parameter Name: bps-dropped
o  CBOR Key Value: 27
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: pkts-dropped
o  CBOR Key Value: 28
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: pps-dropped
o  CBOR Key Value: 29
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: session-id
o  CBOR Key Value: 30
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: trigger-mitigation
o  CBOR Key Value: 31
o  CBOR Major Type: 7
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: missing-hb-allowed
o  CBOR Key Value: 32
o  CBOR Major Type: 5
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: current-value
o  CBOR Key Value: 33
o  CBOR Major Type: 0
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: mitigation-start
o  CBOR Key Value: 34
o  CBOR Major Type: 7
o  Change Controller: IESG
o  Specification Document(s): this document
o  Parameter Name: target-prefix
o  CBOR Key Value: 35
o  CBOR Major Type: 4
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: client-identifier client-domain-hash
o  CBOR Key Value: 36
o  CBOR Major Type: 2 3
o  Change Controller: IESG
o  Specification Document(s): this document


o  Parameter Name: alt-server
```

```
        o  CBOR Key Value: 37
        o  CBOR Major Type: 2
        o  Change Controller: IESG
        o  Specification Document(s): this document


        o  Parameter Name: alt-server-record
        o  CBOR Key Value: 38
        o  CBOR Major Type: 4
        o  Change Controller: IESG
        o  Specification Document(s): this document


        o  Parameter Name: addr
        o  CBOR Key Value: 39
        o  CBOR Major Type: 2
        o  Change Controller: IESG
        o  Specification Document(s): this document


        o  Parameter Name: ttl
        o  CBOR Key Value: 40
        o  CBOR Major Type: 0
        o  Change Controller: IESG
        o  Specification Document(s): this document


        o  Parameter Name: conflict-scope
        o  CBOR Key Value: 41
        o  CBOR Major Type: 5
        o  Change Controller: IESG
        o  Specification Document(s): this document


        o  Parameter Name: acl-name
        o  CBOR Key Value: 42
        o  CBOR Major Type: 3
        o  Change Controller: IESG
        o  Specification Document(s): this document
        o  Parameter Name: acl-type
        o  CBOR Key Value: 43
        o  CBOR Major Type: 3
        o  Change Controller: IESG
        o  Specification Document(s): this document


        o  Parameter Name: config-interval
        o  CBOR Key Value: 44
        o  CBOR Major Type: 0
        o  Change Controller: IESG
        o  Specification Document(s): this document


        o  Parameter Name: attack-time-config mitigating-config
        o  CBOR Key Value: 45
        o  CBOR Major Type: 5
        o  Change Controller: IESG
        o  Specification Document(s): this document


        o  Parameter Name: peace-time-config idle-config
        o  CBOR Key Value: 46
        o  CBOR Major Type: 5
        o  Change Controller: IESG
        o  Specification Document(s): this document

        o  Parameter Name: cuid
        o  CBOR Key Value: 47
        o  CBOR Major Type: 3
        o  Change Controller: IESG
        o  Specification Document(s): this document

        o  Parameter Name: min-value-decimal
        o  CBOR Key Value: 48
        o  CBOR Major Type: 7
        o  Change Controller: IESG
        o  Specification Document(s): this document

        o  Parameter Name: max-value-decimal
        o  CBOR Key Value: 49
        o  CBOR Major Type: 7
        o  Change Controller: IESG
        o  Specification Document(s): this document

        o  Parameter Name: current-value-decimal
        o  CBOR Key Value: 50
        o  CBOR Major Type: 7
        o  Change Controller: IESG
        o  Specification Document(s): this document
```

9.5. DOTS Signal Channel YANG Module

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

           URI: urn:ietf:params:xml:ns:yang:ietf-dots-signal
           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" registry [RFC7950].

           name: ietf-signal
           namespace: urn:ietf:params:xml:ns:yang:ietf-dots-signal
           prefix: signal
           reference: RFC XXXX
```

10. Implementation Status

[Note to RFC Editor: Please remove this section and reference to
[RFC7942] prior to publication.]

This section records the status of known implementations of the
protocol defined by this specification at the time of posting this
Internet-Draft, and is based upon a proposal described in [RFC7942].
The description of implementations in this section is intended to
assist the IETF in its decision-making process when progressing
drafts to RFCs. Please note that the listing of any individual
implementation here does not imply endorsement by the IETF.
Furthermore, no effort has been spent to verify the information
presented here, and which was provided by individuals. This is not
intended as, and must not be construed to be, a catalog of available
implementations or features. Readers are advised to note that other
implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups
to assign due consideration to documents that have the benefit of
running code, which may serve as evidence of valuable experimentation
and feedback that have made the implemented protocols more mature.
It is up to the individual working groups to use this information as
they see fit".

10.1. nttdots

   Organization: NTT Communication is developing a DOTS client and
      DOTS server software based on DOTS signal channel specified in
      this draft. It will be open-sourced.

   Description: Early implementation of DOTS protocol. It is aimed to
      implement a full DOTS protocol specification in accordance with
      the nurturing DOTS protocol.
   Implementation: https://github.com/nttdots/go-dots
   Level of maturity: It is an early implementation of the DOTS
      protocol. Messaging between DOTS clients and DOTS servers has
      been tested. Level of maturity will increase in accordance with
      the nurturing DOTS protocol.
   Coverage: Capability of DOTS client: sending DOTS messages to the
      DOTS server in CoAP over DTLS as dots-signal. Capability of DOTS
      server: receiving dots-signal, validating received dots-signal,
      starting mitigation by handing over the dots-signal to ~~DDOS~~ **DDoS**
      mitigator.
   Licensing: It will be open-sourced with BSD 3-clause license.
   Implementation experience: It is implemented in Go-lang. Core
      specification of signaling is mature to be implemented, however,
      finding good libraries(like DTLS, CoAP) is rather difficult.
   Contact: Kaname Nishizuka <kaname@nttv6.jp>

11. Security Considerations

   Authenticated encryption MUST be used for data confidentiality and
   message integrity. The interaction between the DOTS agents requires
   Datagram Transport Layer Security (DTLS) and Transport Layer Security
   (TLS) with a cipher suite offering confidentiality protection and the
   guidance given in [RFC7525] MUST be followed to avoid attacks on
   (D)TLS. The (D)TLS protocol profile for DOTS signal channel is
   specified in Section 7.

   A single DOTS signal channel between DOTS agents can be used to
   exchange multiple DOTS signal messages. To reduce DOTS client and
   DOTS server workload, DOTS clients SHOULD re-use the (D)TLS session.

   If TCP is used between DOTS agents, an attacker may be able to inject
   RST packets, bogus application segments, etc., regardless of whether
   TLS authentication is used. Because the application data is TLS
   protected, this will not result in the application receiving bogus
   data, but it will constitute a DoS on the connection. This attack
   can be countered by using TCP-AO [RFC5925]. If TCP-AO is used, then
   any bogus packets injected by an attacker will be rejected by the
   TCP-AO integrity check and therefore will never reach the TLS layer.

   In order to prevent leaking internal information outside a client-
   domain, DOTS gateways located in the client-domain SHOULD NOT reveal
   the identification information that pertains to internal DOTS clients
   ~~(client-identifier)~~
   (e.g., source IP address, client's hostname) unless explicitly
   configured to do so.

   Special care should be taken in order to ensure that the activation
   of the proposed mechanism will not impact the stability of the
   network (including connectivity and services delivered over that
   network).

12. Contributors

   The following individuals have contributed to this document:

   Mike Geller Cisco Systems, Inc. 3250 Florida 33309 USA Email:
   mgeller@cisco.com

   Robert Moskowitz HTT Consulting Oak Park, MI 42837 United States
   Email: rgm@htt-consult.com

   Dan Wing Email: dwing-ietf@fuggles.com

13. Acknowledgements

   Thanks to Christian Jacquenet, Roland Dobbins, Roman D. Danyliw,
   Michael Richardson, Ehud Doron, Kaname Nishizuka, Dave Dolson, Liang

Xia, Gilbert Clark, and Nesredien Suleiman for the discussion and
comments.

Special thanks to Jon Shallow for the careful reviews and inputs that
enhanced this specification.

14. References

14.1. Normative References

   [I-D.ietf-core-coap-tcp-tls]
              Bormann, C., Lemay, S., Tschofenig, H., Hartke, K.,
              Silverajan, B., and B. Raymor, "CoAP (Constrained
              Application Protocol) over TCP, TLS, and WebSockets",
              draft-ietf-core-coap-tcp-tls-10
              draft-ietf-core-coap-tcp-tls-11 (work in progress),
              October
              December 2017.

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

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

   [RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key
              Ciphersuites for Transport Layer Security (TLS)",
              RFC 4279, DOI 10.17487/RFC4279, December 2005,
              <https://www.rfc-editor.org/info/rfc4279>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
              Housley, R., and W. Polk, "Internet X.509 Public Key
              Infrastructure Certificate and Certificate Revocation List
              (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
              <https://www.rfc-editor.org/info/rfc5280>.

   [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known
              Uniform Resource Identifiers (URIs)", RFC 5785,
              DOI 10.17487/RFC5785, April 2010,
              <https://www.rfc-editor.org/info/rfc5785>.

   [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP
              Authentication Option", RFC 5925, DOI 10.17487/RFC5925,
              June 2010, <https://www.rfc-editor.org/info/rfc5925>.

   **[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS)
              Extensions: Extension Definitions", RFC 6066,
              DOI 10.17487/RFC6066, January 2011,
              <https://www.rfc-editor.org/info/rfc6066>.**

   [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and
              Verification of Domain-Based Application Service Identity
              within Internet Public Key Infrastructure Using X.509
              (PKIX) Certificates in the Context of Transport Layer
              Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March
              2011, <https://www.rfc-editor.org/info/rfc6125>.

   [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms
              (SHA and SHA-based HMAC and HKDF)", RFC 6234,
              DOI 10.17487/RFC6234, May 2011,
              <https://www.rfc-editor.org/info/rfc6234>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <https://www.rfc-editor.org/info/rfc6347>.

   [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J.,
              Weiler, S., and T. Kivinen, "Using Raw Public Keys in
              Transport Layer Security (TLS) and Datagram Transport
              Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250,
              June 2014, <https://www.rfc-editor.org/info/rfc7250>.

   [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained
              Application Protocol (CoAP)", RFC 7252,
              DOI 10.17487/RFC7252, June 2014,
              <https://www.rfc-editor.org/info/rfc7252>.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May
              2015, <https://www.rfc-editor.org/info/rfc7525>.

   [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>.

    [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
               Writing an IANA Considerations Section in RFCs", BCP 26,
               RFC 8126, DOI 10.17487/RFC8126, June 2017,
               <https://www.rfc-editor.org/info/rfc8126>.

    [RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and
               FETCH Methods for the Constrained Application Protocol
               (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017,
               <https://www.rfc-editor.org/info/rfc8132>.

14.2. Informative References

    [I-D.ietf-core-comi]
               Veillette, M., Stok, P., Pelov, A., and A. Bierman, "CoAP
               Management Interface", draft-ietf-core-comi-02 (work in
               progress), December 2017.

    [I-D.ietf-core-yang-cbor]
               Veillette, M., Pelov, A., Somaraju, A., Turner, R., and A.
               Minaburo, "CBOR Encoding of Data Modeled with YANG",
               draft-ietf-core-yang-cbor-05 (work in progress), August
               2017.

    [I-D.ietf-dots-architecture]
               Mortensen, A., Andreasen, F., Reddy, T.,
               christopher_gray3@cable.comcast.com, c., Compton, R., and
               N. Teague, "Distributed-Denial-of-Service Open Threat
               Signaling (DOTS) Architecture", draft-ietf-dots-
               architecture-05 (work in progress), October 2017.

    [I-D.ietf-dots-data-channel]
               Reddy, T., Boucadair, M., Nishizuka, K., Xia, L., Patil,
               P., Mortensen, A., and N. Teague, "Distributed Denial-of-
               Service Open Threat Signaling (DOTS) Data Channel", draft-
               ietf-dots-data-channel-10
               ietf-dots-data-channel-11 (work in progress), December
               2017.

    [I-D.ietf-dots-requirements]
               Mortensen, A., Moskowitz, R., and T. Reddy, "Distributed
               Denial of Service (DDoS) Open Threat Signaling
               Requirements", draft-ietf-dots-requirements-08 draft-ietf-dots-requirements-10 (work in
               progress), December 2017. January 2018.

    [I-D.ietf-dots-use-cases]
               Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
               Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
               Open Threat Signaling", draft-ietf-dots-use-cases-09 (work
               in progress), November 2017.

    [I-D.ietf-netmod-yang-tree-diagrams]
               Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-
               ietf-netmod-yang-tree-diagrams-02
               ietf-netmod-yang-tree-diagrams-04 (work in progress),
               October
               December 2017.

    [I-D.ietf-tls-dtls13]
               Rescorla, E., Tschofenig, H., and N. Modadugu, "The
               Datagram Transport Layer Security (DTLS) Protocol Version
               1.3", draft-ietf-tls-dtls13-22 (work in progress),
               November 2017.

    [I-D.ietf-tls-tls13]
               Rescorla, E., "The Transport Layer Security (TLS) Protocol
               Version 1.3", draft-ietf-tls-tls13-22 draft-ietf-tls-tls13-23 (work in progress),
               November 2017.
               January 2018.

    [proto_numbers]
               "IANA, "Protocol Numbers"", 2011,
               <http://www.iana.org/assignments/protocol-numbers>.

    [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791,
               DOI 10.17487/RFC0791, September 1981,
               <https://www.rfc-editor.org/info/rfc791>.

    [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network
               Address Translator (Traditional NAT)", RFC 3022,
               DOI 10.17487/RFC3022, January 2001,
               <https://www.rfc-editor.org/info/rfc3022>.

    [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
               Resource Identifier (URI): Generic Syntax", STD 66,
               RFC 3986, DOI 10.17487/RFC3986, January 2005,
               <https://www.rfc-editor.org/info/rfc3986>.

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

    [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram
               Congestion Control Protocol (DCCP)", RFC 4340,
               DOI 10.17487/RFC4340, March 2006,
               <https://www.rfc-editor.org/info/rfc4340>.

    [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing
               (CIDR): The Internet Address Assignment and Aggregation
               Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August

2006, <https://www.rfc-editor.org/info/rfc4632>.

[RFC4732]   Handley, M., Ed., Rescrla, E., Ed., and IAB, "Internet
            Denial-of-Service Considerations", RFC 4732,
            DOI 10.17487/RFC4732, December 2006,
            <https://www.rfc-editor.org/info/rfc4732>.

[RFC4787]   Audet, F., Ed. and C. Jennings, "Network Address
            Translation (NAT) Behavioral Requirements for Unicast
            UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January
            2007, <https://www.rfc-editor.org/info/rfc4787>.

[RFC4941]   Narten, T., Draves, R., and S. Krishnan, "Privacy
            Extensions for Stateless Address Autoconfiguration in
            IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007,
            <https://www.rfc-editor.org/info/rfc4941>.

[RFC4960]   Stewart, R., Ed., "Stream Control Transmission Protocol",
            RFC 4960, DOI 10.17487/RFC4960, September 2007,
            <https://www.rfc-editor.org/info/rfc4960>.

[RFC4987]   Eddy, W., "TCP SYN Flooding Attacks and Common
            Mitigations", RFC 4987, DOI 10.17487/RFC4987, August 2007,
            <https://www.rfc-editor.org/info/rfc4987>.

[RFC5077]   Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig,
            "Transport Layer Security (TLS) Session Resumption without
            Server-Side State", RFC 5077, DOI 10.17487/RFC5077,
            January 2008, <https://www.rfc-editor.org/info/rfc5077>.

[RFC5389]   Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
            "Session Traversal Utilities for NAT (STUN)", RFC 5389,
            DOI 10.17487/RFC5389, October 2008,
            <https://www.rfc-editor.org/info/rfc5389>.

[RFC6146]   Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful
            NAT64: Network Address and Protocol Translation from IPv6
            Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146,
            April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6296]   Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix
            Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011,
            <https://www.rfc-editor.org/info/rfc6296>.

[RFC6555]   Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with
            Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April
            2012, <https://www.rfc-editor.org/info/rfc6555>.

[RFC6724]   Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown,
            "Default Address Selection for Internet Protocol Version 6
            (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012,
            <https://www.rfc-editor.org/info/rfc6724>.

[RFC6887]   Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and
            P. Selkirk, "Port Control Protocol (PCP)", RFC 6887,
            DOI 10.17487/RFC6887, April 2013,
            <https://www.rfc-editor.org/info/rfc6887>.

[RFC6888]   Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa,
            A., and H. Ashida, "Common Requirements for Carrier-Grade
            NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888,
            April 2013, <https://www.rfc-editor.org/info/rfc6888>.

[RFC7030]   Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
            "Enrollment over Secure Transport", RFC 7030,
            DOI 10.17487/RFC7030, October 2013,
            <https://www.rfc-editor.org/info/rfc7030>.

[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>.

[RFC7413]   Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP
            Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014,
            <https://www.rfc-editor.org/info/rfc7413>.

[RFC7452]   Tschofenig, H., Arkko, J., Thaler, D., and D. McPherson,
            "Architectural Considerations in Smart Object Networking",
            RFC 7452, DOI 10.17487/RFC7452, March 2015,
            <https://www.rfc-editor.org/info/rfc7452>.

[RFC7589]   Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the
            NETCONF Protocol over Transport Layer Security (TLS) with
            Mutual X.509 Authentication", RFC 7589,
            DOI 10.17487/RFC7589, June 2015,
            <https://www.rfc-editor.org/info/rfc7589>.

[RFC7918]   Langley, A., Modadugu, N., and B. Moeller, "Transport
            Layer Security (TLS) False Start", RFC 7918,
            DOI 10.17487/RFC7918, August 2016,
            <https://www.rfc-editor.org/info/rfc7918>.

[RFC7924]   Santesson, S. and H. Tschofenig, "Transport Layer Security
            (TLS) Cached Information Extension", RFC 7924,
            DOI 10.17487/RFC7924, July 2016,
            <https://www.rfc-editor.org/info/rfc7924>.

[RFC7942]   Sheffer, Y. and A. Farrel, "Improving Awareness of Running
            Code: The Implementation Status Section", BCP 205,
            RFC 7942, DOI 10.17487/RFC7942, July 2016,

                  <https://www.rfc-editor.org/info/rfc7942>.

   [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG",
              RFC 7951, DOI 10.17487/RFC7951, August 2016,
              <https://www.rfc-editor.org/info/rfc7951>.

   [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage
              Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085,
              March 2017, <https://www.rfc-editor.org/info/rfc8085>.

Authors' Addresses
   Tirumaleswar Reddy
   McAfee, Inc.
   Embassy Golf Link Business Park
   Bangalore, Karnataka   560071
   India

   Email: kondtir@gmail.com

   Mohamed Boucadair
   Orange
   Rennes   35000
   France

   Email: mohamed.boucadair@orange.com

   Prashanth Patil
   Cisco Systems, Inc.

   Email: praspati@cisco.com

   Andrew Mortensen
   Arbor Networks, Inc.
   2727 S. State St
   Ann Arbor, MI   48104
   United States

   Email: amortensen@arbor.net

   Nik Teague
   Verisign, Inc.
   United States

   Email: nteague@verisign.com