Workgroup: NETCONF
Internet-Draft:
draft-ietf-netconf-udp-notif-09
Published: 10 March 2023
Intended Status: Standards Track
Expires: 11 September 2023
Authors: G. Zheng (Huawei)
T. Zhou (Huawei)
T. Graf (Swisscom)
P. Francois (INSA-Lyon)
A. Huang Feng (INSA-Lyon)
P. Lucente (NTT)

# UDP-based Transport for Configured Subscriptions

## Abstract

This document describes an UDP-based notification mechanism to collect data from networking devices. A shim header is proposed to facilitate the data streaming directly from the publishing process on network processor of line cards to receivers. The objective is to provide a lightweight approach to enable higher frequency and less performance impact on publisher and receiver processes compared to already established notification mechanisms.

## Requirements Language

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

## 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 11 September 2023.

## Copyright Notice



## Table of Contents

## 1. Introduction

Sub-Notif [RFC8639] defines a mechanism that lets a receiver subscribe to the publication of YANG-defined data maintained in a YANG [RFC7950] datastore. The mechanism separates the management and control of subscriptions from the transport used to deliver the data. Three transport mechanisms, namely NETCONF transport [RFC8640], RESTCONF transport [RFC8650], and HTTPS transport [I-D.ietf-netconf-https-notif] have been defined so far for such notification messages.

While powerful in their features and general in their architecture, the currently available transport mechanisms need to be complemented to support data publications at high velocity from devices that feature a distributed architecture. The currently available transports are based on TCP and lack the efficiency needed to continuously send notifications at high velocity.

This document specifies a transport option for Sub-Notif that leverages UDP. Specifically, it facilitates the distributed data collection mechanism described in [I-D.ietf-netconf-distributed-notif]. In the case of publishing from multiple network processors on multiple line cards, centralized designs require data to be internally forwarded from those network processors to the push server, presumably on a route processor, which then combines the individual data items into a single consolidated stream. The centralized data collection mechanism can result in a performance bottleneck, especially when large amounts of data are involved.

What is needed is a mechanism that allows for directly publishing from multiple network processors on line cards, without passing them through an additional processing stage for internal consolidation. The proposed UDP-based transport allows for such a distributed data publishing approach.

* Firstly, a UDP approach reduces the burden of maintaining a large amount of active TCP connections at the receiver, notably in cases where it collects data from network processors on line cards from a large amount of networking devices.

* Secondly, as no connection state needs to be maintained, UDP encapsulation can be easily implemented by the hardware of the publication streamer, which will further improve performance.

*Ultimately, such advantages allow for a larger data analysis feature set, as more voluminous, finer grained data sets can be streamed to the receiver.

The transport described in this document can be used for transmitting notification messages over both IPv4 and IPv6.

This document describes the notification mechanism. It is intended to be used in conjunction with [RFC8639], extended by [I-D.ietf-netconf-distributed-notif].

Section 2 describes the control of the proposed transport mechanism. Section 3 details the notification mechanism and message format. Section 4 describes the use of options in the notification message header. Section 5 covers the applicability of the proposed mechanism. Section 6 describes a mechanism to secure the protocol in open networks.

## 2. Configured Subscription to UDP-Notif

This section describes how the proposed mechanism can be controlled using subscription channels based on NETCONF or RESTCONF.

Following the usual approach of Sub-Notif, configured subscriptions contain the location information of all the receivers, including the IP address and the port number, so that the publisher can actively send UDP-Notif messages to the corresponding receivers.

Note that receivers MAY NOT be already up and running when the configuration of the subscription takes effect on the monitored device. The first message MUST be a separate subscription-started notification to indicate the Receiver that the stream has started flowing. Then, the notifications can be sent immediately without delay. All the subscription state notifications, as defined in [RFC8639], MUST be encapsulated in separate notification messages.

## 3. UDP-Based Transport

In this section, we specify the UDP-Notif Transport behavior. Section 3.1 describes the general design of the solution. Section 3.2 specifies the UDP-Notif message format. Section 4 describes a generic optional sub TLV format. Section 4.1 uses such options to provide a segmentation solution for large UDP-Notif message payloads. Section 3.3 describes the encoding of the message payload.

### 3.1. Design Overview

As specified in Sub-Notif, the telemetry data is encapsulated in the NETCONF/RESTCONF notification message, which is then encapsulated

and carried using transport protocols such as TLS or HTTP2. This document defines a UDP based transport. Figure 1 illustrates the structure of an UDP-Notif message.

*The Message Header contains information that facilitate the message transmission before deserializing the notification message.

*Notification Message is the encoded content that the publication stream transports. The common encoding methods are listed in Section 3.2. The structure of the Notification Message is defined in Section 2.6 of [RFC8639].
[I-D.ietf-netconf-notification-messages] proposes a structure to send bundled notifications in a single message.


Figure 1: UDP-Notif Message Overview

### 3.2. Format of the UDP-Notif Message Header

The UDP-Notif Message Header contains information that facilitate the message transmission before deserializing the notification message. The data format is shown in Figure 2.

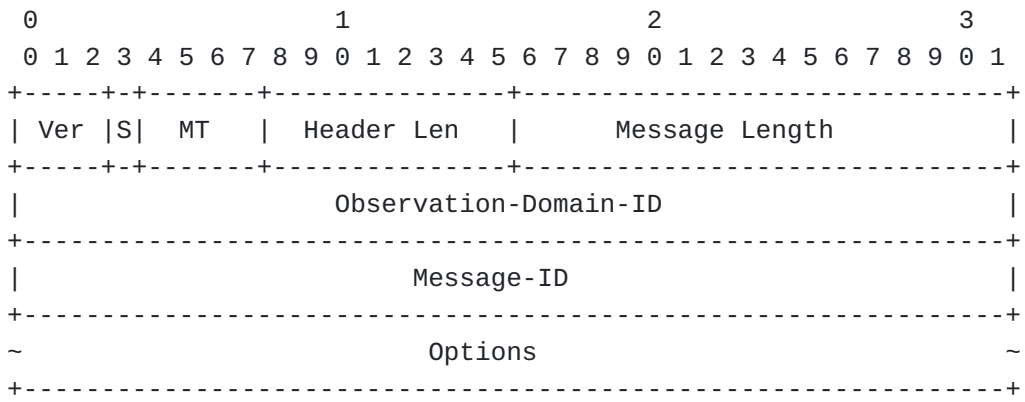


Figure 2: UDP-Notif Message Header Format

The Message Header contains the following field:

*Ver represents the PDU (Protocol Data Unit) encoding version. The current version value is 1.

*S represents the space of media type specified in the MT field. When S is unset, MT represents the standard media types as defined in this document. When S is set, MT represents a private space to be freely used for non standard encodings.

*MT is a 4 bit identifier to indicate the media type used for the Notification Message. 16 types of encoding can be expressed. When the S bit is unset, the following values apply:

  -0: Reserved;

  -1: application/yang-data+json [RFC8040]

  -2: application/yang-data+xml [RFC8040]

  -3: application/yang-data+cbor [RFC9254]

*Header Len is the length of the message header in octets, including both the fixed header and the options.

*Message Length is the total length of the message within one UDP datagram, measured in octets, including the message header.

*Observation-Domain-ID is a 32-bit identifier of the Observation Domain that led to the production of the notification message, as defined in [I-D.ietf-netconf-distributed-notif]. This allows disambiguation of an information source, such as the identification of different line cards sending the notification messages. Message unicity is obtained from the conjonction of the Observation-Domain-ID and the Message-ID field described below. If Observation-Domain-ID unicity is not preserved through the collection domain, the source IP address of the UDP datagram SHOULD be used in addition to the Observation-Domain-ID to identify the information source. If a transport layer relay is used, Observation-Domain-ID unicity must be preserved through the collection domain.

*The Message-ID is generated continuously by the publisher of UDP-Notif messages. A publisher MUST use different Message-ID values for different messages generated with the same Observation-Domain-ID. Note that the main purpose of the Message-ID is to reconstruct messages which were segmented using the segmentation option described in section Section 4.1. The Message-ID values SHOULD be incremented by one for each successive message originated with the same Observation-Domain-ID, so that message loss can be detected. Furthermore, incrementing the Message-ID by one allows for a large amount of time to happen before the Message-ID's are reused due to wrapping around. Different subscribers MAY share the same Message-ID sequence.

*Options is a variable-length field in the TLV format. When the Header Length is larger than 12 octets, which is the length of the fixed header, Options TLVs follow directly after the fixed message header (i.e., Message-ID). The details of the options are described in Section 4.

### 3.3. Data Encoding

UDP-Notif message data can be encoded in CBOR, XML or JSON format. It is conceivable that additional encodings may be supported in the future. This can be accomplished by augmenting the subscription data model with additional identity statements used to refer to requested encodings.

Private encodings can be supported through the use of the S bit of the header. When the S bit is set, the value of the MT field is left to be defined and agreed upon by the users of the private encoding. An option is defined in Section 4.2 for more verbose encoding descriptions than what can be described with the MT field.

Implementation MAY support multiple encoding methods per subscription. When bundled notifications are supported between the publisher and the receiver, only subscribed notifications with the same encoding can be bundled in a given message.

## 4. Options

All the options are defined with the following format, illustrated in Figure 3.

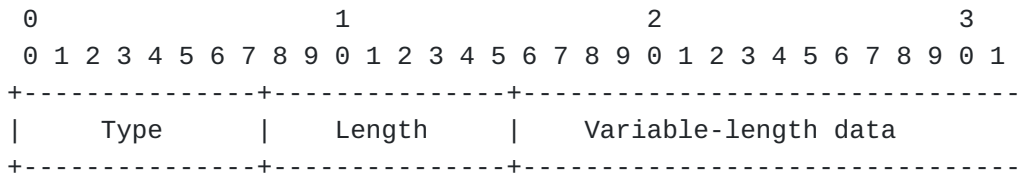


Figure 3: Generic Option Format

*Type: 1 octet describing the option type;

*Length: 1 octet representing the total number of octets in the TLV, including the Type and Length fields;

*Variable-length data: 0 or more octets of TLV Value.

When more than one option are used in the UDP-notif header, options MUST be ordered by the Type value. Messages with unordered options MAY be dropped by the Receiver.

### 4.1. Segmentation Option

The UDP payload length is limited to 65535. Application level headers will make the actual payload shorter. Even though binary encodings such as CBOR may not require more space than what is left, more voluminous encodings such as JSON and XML may suffer from this size limitation. Although IPv4 and IPv6 publishers can fragment outgoing packets exceeding their Maximum Transmission Unit(MTU), fragmented IP packets may not be desired for operational and performance reasons.

Consequently, implementations of the mechanism SHOULD provide a configurable max-segment-size option to control the maximum size of a payload.

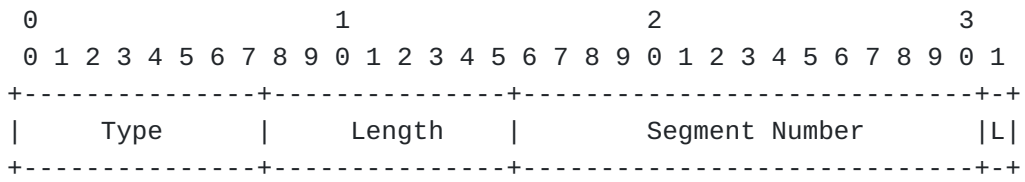


Figure 4: Segmentation Option Format

The Segmentation Option is to be included when the message content is segmented into multiple pieces. Different segments of one message share the same Message-ID. An illustration is provided in Figure 4. The fields of this TLV are:

- Type: Generic option field which indicates a Segmentation Option. The Type value is to be assigned TBD1.

- Length: Generic option field which indicates the length of this option. It is a fixed value of 4 octets for the Segmentation Option.

- Segment Number: 15-bit value indicating the sequence number of the current segment. The first segment of a segmented message has a Segment Number value of 0.

- L: is a flag to indicate whether the current segment is the last one of the message. When 0 is set, the current segment is not the last one. When 1 is set, the current segment is the last one, meaning that the total number of segments used to transport this message is the value of the current Segment Number + 1.

An implementation of this specification MUST NOT rely on IP fragmentation by default to carry large messages. An implementation of this specification MUST either restrict the size of individual

messages carried over this protocol, or support the segmentation option.

When a message has multiple options and is segmented using the described mechanism, all the options MUST be present on the first segment ordered by the options Type. The rest of segmented messages MAY include all the options ordered by options type.

### 4.2. Private Encoding Option

The space to describe private encodings in the MT field of the UDP-Notif header being limited, an option is provided to describe custom encodings. The fields of this option are as follows.

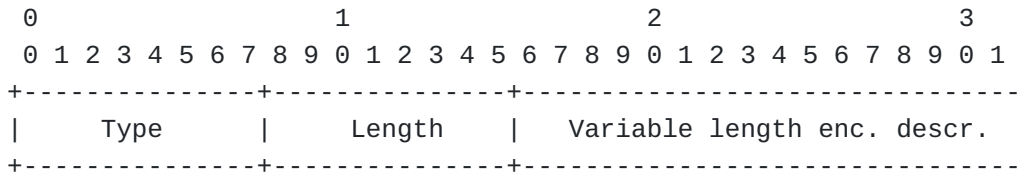


Figure 5: Private Encoding Option Format

*Type: Generic option field which indicates a Private Encoding Option. The Type value is to be assigned TBD2.

*Length: Generic option field which indicates the length of this option. It is a variable value.

*Enc. Descr: The description of the private encoding used for this message. The values to be used for such private encodings is left to be defined by the users of private encodings.

This option SHOULD only be used when the S bit of the header is set, as providing a private encoding description for standard encodings is meaningless.

## 5. Applicability

In this section, we provide an applicability statement for the proposed mechanism, following the recommendations of [RFC8085].

The proposed mechanism falls in the category of UDP applications "designed for use within the network of a single network operator or on networks of an adjacent set of cooperating network operators, to be deployed in controlled environments", as defined in [RFC8085]. Implementations of the proposed mechanism SHOULD thus follow the recommendations in place for such specific applications. In the following, we discuss recommendations on congestion control, message

size guidelines, reliability considerations and security considerations.

### 5.1. Congestion Control

The proposed application falls into the category of applications performing transfer of large amounts of data. It is expected that the operator using the solution configures QoS on its related flows. As per [RFC8085], such applications MAY choose not to implement any form of congestion control, but follow the following principles.

It is NOT RECOMMENDED to use the proposed mechanism over congestion-sensitive network paths. The only environments where UDP-Notif is expected to be used are managed networks. The deployments require that the network path has been explicitly provisioned to handle the traffic through traffic engineering mechanisms, such as rate limiting or capacity reservations.

Implementation of the proposal SHOULD NOT push unlimited amounts of traffic by default, and SHOULD require the users to explicitly configure such a mode of operation.

Burst mitigation through packet pacing is RECOMMENDED. Disabling burst mitigation SHOULD require the users to explicitly configure such a mode of operation.

Applications SHOULD monitor packet losses and provide means to the user for retrieving information on such losses. The UDP-Notif Message ID can be used to deduce congestion based on packet loss detection. Hence the receiver can notify the device to use a lower streaming rate. The interaction to control the streaming rate on the device is out of the scope of this document.

### 5.2. Message Size

[RFC8085] recommends not to rely on IP fragmentation for messages whose size result in IP packets exceeding the MTU along the path. The segmentation option of the current specification permits segmentation of the UDP Notif message content without relying on IP fragmentation. Implementation of the current specification SHOULD allow for the configuration of the MTU.

### 5.3. Reliability

The target application for UDP-Notif is the collection of data-plane information. The lack of reliability of the data streaming mechanism is thus considered acceptable as the mechanism is to be used in controlled environments, mitigating the risk of information loss, while allowing for publication of very large amounts of data. Moreover, in this context, sporadic events when incomplete data

collection is provided is not critical for the proper management of the network, as information collected for the devices through the means of the proposed mechanism is to be often refreshed.

A receiver implementation for this protocol SHOULD deal with potential loss of packets carrying a part of segmented payload, by discarding packets that were received, but cannot be re-assembled as a complete message within a given amount of time. This time SHOULD be configurable.

## 6. Secured layer for UDP-notif

In open or unsecured networks, UDP-notif messages MUST be secured or encrypted. In this section, a mechanism using DTLS 1.3 to secure UDP-notif protocol is presented. The following sections defines the requirements for the implementation of the secured layer of DTLS for UDP-notif. No DTLS 1.3 extensions are defined nor needed.

The DTLS 1.3 protocol [RFC9147] is designed to meet the requirements of applications that need to secure datagram transport. Implementations using DTLS to secure UDP-notif messages MUST use DTLS 1.3 protocol as defined in [RFC9147] without any new extensions.

When this security layer is used, the Publisher MUST always be a DTLS client, and the Receiver MUST always be a DTLS server. The Receivers MUST support accepting UDP-notif Messages on the specified UDP port, but MAY be configurable to listen on a different port. The Publisher MUST support sending UDP-notif messages to the specified UDP port, but MAY be configurable to send messages to a different port. The Publisher MAY use any source UDP port for transmitting messages.

### 6.1. Session lifecycle

#### 6.1.1. DTLS Session Initiation

The Publisher initiates a DTLS connection by sending a DTLS ClientHello to the Receiver. Implementations MAY support the denial of service countermeasures defined by DTLS 1.3. When these countermeasures are used, the Receiver responds with a DTLS HelloRetryRequest containing a stateless cookie. The Publisher MUST send a new DTLS ClientHello message containing the received cookie, which initiates the DTLS handshake.

When DTLS is implemented, the Publisher MUST NOT send any UDP-notif messages before the DTLS handshake has successfully completed. Early data mechanism (also known as 0-RTT data) as defined in [RFC9147] MUST NOT be used.

Implementations of this security layer MUST support DTLS 1.3 [RFC9147] and MUST support the mandatory to implement cipher suite TLS_AES_128_GCM_SHA256 and SHOULD implement TLS_AES_256_GCM_SHA384 and TLS_CHACHA20_POLY1305_SHA256 cipher suites, as specified in TLS 1.3 [RFC8446]. If additional cipher suites are supported, then implementations MUST NOT negotiate a cipher suite that employs NULL integrity or authentication algorithms.

Where privacy is REQUIRED, then implementations must either negotiate a cipher suite that employs a non-NULL encryption algorithm or otherwise achieve privacy by other means, such as a physically secured network.

#### 6.1.2. Publish Data

When DTLS is used, all UDP-notif messages MUST be published as DTLS "application_data". It is possible that multiple UDP-notif messages are contained in one DTLS record, or that a publication message is transferred in multiple DTLS records. The application data is defined with the following ABNF [RFC5234] expression:

```
APPLICATION-DATA = 1*UDP-NOTIF-FRAME

UDP-NOTIF-FRAME = MSG-LEN SP UDP-NOTIF-MSG

MSG-LEN = NONZERO-DIGIT *DIGIT

SP = %d32

NONZERO-DIGIT = %d49-57

DIGIT = %d48 / NONZERO-DIGIT
```

UDP-NOTIF-MSG is defined in Section 3.

The Publisher SHOULD attempt to avoid IP fragmentation by using the Segmentation Option in the UDP-notif message.

#### 6.1.3. Session termination

A Publisher MUST close the associated DTLS connection if the connection is not expected to deliver any UDP-notif Messages later. It MUST send a DTLS close_notify alert before closing the connection. A Publisher (DTLS client) MAY choose to not wait for the Receiver's close_notify alert and simply close the DTLS connection. Once the Receiver gets a close_notify from the Publisher, it MUST reply with a close_notify.

When no data is received from a DTLS connection for a long time, the Receiver MAY close the connection. Implementations SHOULD set the

timeout value to 10 minutes but application specific profiles MAY recommend shorter or longer values. The Receiver (DTLS server) MUST attempt to initiate an exchange of close_notify alerts with the Publisher before closing the connection. Receivers that are unprepared to receive any more data MAY close the connection after sending the close_notify alert.

Although closure alerts are a component of TLS and so of DTLS, they, like all alerts, are not retransmitted by DTLS and so may be lost over an unreliable network.

## 7. A YANG Data Model for Management of UDP-Notif

The YANG model described in Section 8 defines a new receiver instance for UDP-notif transport. When this transport is used, four new leaves and a dtls container allow configuring UDP-notif receiver parameters.

```
module: ietf-udp-notif-transport

  augment /sn:subscriptions/snr:receiver-instances
            /snr:receiver-instance/snr:transport-type:
    +--:(udp-notif)
       +--rw udp-notif-receiver
          +--rw address                inet:ip-address-no-zone
          +--rw port                   inet:port-number
          +--rw enable-segmentation?   boolean {segmentation}?
          +--rw max-segment-size?      uint32 {segmentation}?
          +--rw dtls! {dtls-supported}?
             +--rw client-identity!
             |  +--rw (auth-type)
             |     +--:(certificate) {client-ident-x509-cert}?
             |     |     ...
             |     +--:(raw-public-key) {client-ident-raw-public-key}?
             |     |     ...
             |     +--:(tls13-epsk) {client-ident-tls13-epsk}?
             |           ...
             +--rw server-authentication
             |  +--rw ca-certs! {server-auth-x509-cert}?
             |  |  +--rw (local-or-truststore)
             |  |        ...
             |  +--rw ee-certs! {server-auth-x509-cert}?
             |  |  +--rw (local-or-truststore)
             |  |        ...
             |  +--rw raw-public-keys! {server-auth-raw-public-key}?
             |  |  +--rw (local-or-truststore)
             |  |        ...
             |  +--rw tls13-epsks?       empty
             |           {server-auth-tls13-epsk}?
             +--rw hello-params {tlscmn:hello-params}?
             |  +--rw tls-versions
             |  |  +--rw tls-version*   identityref
             |  +--rw cipher-suites
             |     +--rw cipher-suite*   identityref
             +--rw keepalives {tls-client-keepalives}?
                +--rw peer-allowed-to-send?   empty
                +--rw test-peer-aliveness!
                   +--rw max-wait?       uint16
                   +--rw max-attempts?   uint8
```

## 8. YANG Module

This YANG module is used to configure, on a publisher, a receiver willing to consume notification messages. This module augments the "ietf-subscribed-notif-receivers" module to define a UDP-notif transport receiver. The grouping "udp-notif-receiver-grouping" defines the necessary parameters to configure this transport. The

grouping uses tls-client-grouping defined in
[I-D.ietf-netconf-tls-client-server] to add DTLS 1.3 parameters.

```
<CODE BEGINS> file "ietf-udp-notif-transport@2023-03-10.yang"

module ietf-udp-notif-transport {
  yang-version 1.1;
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport";
  prefix unt;
  import ietf-subscribed-notifications {
    prefix sn;
    reference
      "RFC 8639: Subscription to YANG Notifications";
  }
  import ietf-subscribed-notif-receivers {
    prefix snr;
    reference
      "RFC YYYY: An HTTPS-based Transport for
                 Configured Subscriptions";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC TTTT: YANG Groupings for TLS Clients and TLS Servers";
  }

  organization "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   <http:/tools.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Authors:  Guangying Zheng
               <mailto:zhengguangying@huawei.com>
               Tianran Zhou
               <mailto:zhoutianran@huawei.com>
               Thomas Graf
               <mailto:thomas.graf@swisscom.com>
               Pierre Francois
               <mailto:pierre.francois@insa-lyon.fr>
               Alex Huang Feng
               <mailto:alex.huang-feng@insa-lyon.fr>
               Paolo Lucente
               <mailto:paolo@ntt.net>";

  description
    "Defines UDP-Notif as a supported transport for subscribed
```

```
    event notifications.

    Copyright (c) 2022 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 Revised BSD License set forth in Section
    4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
    (https://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 2023-03-10 {
    description
      "Initial revision";
    reference
      "RFC XXXX: UDP-based Transport for Configured Subscriptions";
  }

 /*
  * FEATURES
  */
  feature encode-cbor {
    description
      "This feature indicates that CBOR encoding of notification
       messages is supported.";
  }
  feature dtls-supported {
    description
      "This feature indicates that DTLS encryption of notification
       messages is supported.";
  }
  feature segmentation {
    description
      "This feature indicates segmentation of notification messages
      is supported.";
  }

 /*
  * IDENTITIES
  */
  identity udp-notif {
    base sn:transport;
    description
      "UDP-Notif is used as transport for notification messages
        and state change notifications.";
  }
```

```
identity encode-cbor {
  base sn:encoding;
  description
    "Encode data using CBOR as described in RFC 9254.";
  reference
    "RFC 9254: CBOR Encoding of Data Modeled with YANG";
}

grouping udp-notif-receiver-grouping {
  description
    "Provides a reusable description of a UDP-Notif target
    receiver.";

  leaf address {
    type inet:ip-address-no-zone;
    mandatory true;
    description
      "IP address of target UDP-Notif receiver, which can be an
      IPv4 address or an IPV6 address.";
  }

  leaf port {
    type inet:port-number;
    mandatory true;
    description
      "Port number of target UDP-Notif receiver.";
  }

  leaf enable-segmentation {
    if-feature segmentation;
    type boolean;
    default false;
    description
      "The switch for the segmentation feature. When disabled, the
      publisher will not allow fragment for a very large data";
  }

  leaf max-segment-size {
    when "../enable-segmentation = 'true'";
    if-feature segmentation;
    type uint32;
    description
      "UDP-Notif provides a configurable max-segment-size to
      control the size of each segment (UDP-Notif header, with
      options, included).";
  }

  container dtls {
```

```
      if-feature dtls-supported;
      presence dtls;
      uses tlsc:tls-client-grouping {
        // Using tls-client-grouping without TLS1.2 parameters
        // allowing only DTLS 1.3
        refine "client-identity/auth-type/tls12-psk" {
          // create the logical impossibility of enabling TLS1.2
          if-feature "not tlsc:client-ident-tls12-psk";
        }
        refine "server-authentication/tls12-psks" {
          // create the logical impossibility of enabling TLS1.2
          if-feature "not tlsc:server-auth-tls12-psk";
        }
      }
      description
        "Container for configuring DTLS 1.3 parameters if DTLS is enable
    }
  }

  augment "/sn:subscriptions/snr:receiver-instances/" +
          "snr:receiver-instance/snr:transport-type" {
    case udp-notif {
      container udp-notif-receiver {
        description
          "The UDP-notif receiver to send notifications to.";
        uses udp-notif-receiver-grouping;
      }
    }
    description
      "Augment the transport-type choice to include the 'udp-notif'
       transport.";
  }
}

<CODE ENDS>
```

## 9. IANA Considerations

This document describes a number of new registries, the URI from IETF XML Registry and the registration of a new YANG module name.

### 9.1. IANA registries

This document is creating 3 registries called "UDP-notif media types", "UDP-notif option types", and "UDP-notif header version" under the new group "UDP-notif protocol". The registration procedure is made using the Standards Action process defined in [RFC8126].

The first requested registry is the following:

Registry Name: UDP-notif media types
Registry Category: UDP-notif protocol.
Registration Procedure: Standard Action as defined in RFC8126
Maximum value: 15

These are the initial registrations for "UDP-notif media types":

Value: 0
Description: Reserved
Reference: this document

Value: 1
Description: media type application/yang-data+json
Reference: <xref target="RFC8040"/>

Value: 2
Description: media type application/yang-data+xml
Reference: <xref target="RFC8040"/>

Value: 3
Description: media type application/yang-data+cbor
Reference: <xref target="RFC9254"/>

The second requested registry is the following:

Registry Name: UDP-notif option types
Registry Category: UDP-notif protocol.
Registration Procedure: Standard Action as defined in RFC8126
Maximum value: 255

These are the initial registrations for "UDP-notif options types":

Value: 0
Description: Reserved
Reference: this document

Value: TBD1 (suggested value: 1)
Description: Segmentation Option
Reference: this document

Value: TBD2 (suggested value: 2)
Description: Private Encoding Option
Reference: this document

The third requested registry is the following:

Registry Name: UDP-notif header version
Registry Category: UDP-notif protocol.
Registration Procedure: Standard Action as defined in RFC8126
Maximum value: 7

These are the initial registrations for "UDP-notif header version":

Value: 0
Description: First version
Reference: draft-ietf-netconf-udp-pub-channel-05

Value: 1
Description: RFCXXXX, current version.
Reference: this document

### 9.2. URI

IANA is also requested to assign a new URI from the IETF XML Registry [RFC3688]. The following URI is suggested:

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

### 9.3. YANG module name

This document also requests a new YANG module name in the YANG Module Names registry [RFC7950] with the following suggestion:

name: ietf-udp-notif
namespace: urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport
prefix: unt
reference: RFC XXXX

## 10. Security Considerations

[RFC8085] states that "UDP applications that need to protect their communications against eavesdropping, tampering, or message forgery SHOULD employ end-to-end security services provided by other IETF protocols". As mentioned above, the proposed mechanism is designed

to be used in controlled environments, as defined in [RFC8085] also known as "limited domains", as defined in [RFC8799]. Thus, a security layer is not necessary required. Nevertheless, a DTLS layer MUST be implemented in open or unsecured networks. A specification of udp-notif using DTLS is presented in Section 6.

## 11. Acknowledgements

The authors of this documents would like to thank Alexander Clemm, Benoit Claise, Eric Voit, Huiyang Yang, Kent Watsen, Mahesh Jethanandani, Marco Tollini, Rob Wilton, Sean Turner, Stephane Frenot, Timothy Carey, Tim Jenkins, Tom Petch and Yunan Gu for their constructive suggestions for improving this document.

## 12. References

### 12.1. Normative References

**[I-D.ietf-netconf-https-notif]** Jethanandani, M. and K. Watsen, "An HTTPS-based Transport for YANG Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-https-notif-13, 4 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-https-notif-13>.

**[I-D.ietf-netconf-tls-client-server]** Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-29, 18 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-29>.

**[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>.

**[RFC5234]** Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI

10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

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

**[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>.

**[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>.

**[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>.

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

**[RFC8446]** Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

**[RFC8639]** Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

**[RFC8640]** Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Dynamic Subscription to YANG Events and Datastores over NETCONF", RFC 8640, DOI 10.17487/RFC8640, September 2019, <https://www.rfc-editor.org/info/rfc8640>.

**[RFC8650]** Voit, E., Rahman, R., Nilsen-Nygaard, E., Clemm, A., and A. Bierman, "Dynamic Subscription to YANG Events and Datastores over RESTCONF", RFC 8650, DOI 10.17487/RFC8650, November 2019, <https://www.rfc-editor.org/info/rfc8650>.

**[RFC9254]** Veillette, M., Ed., Petrov, I., Ed., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", RFC 9254, DOI 10.17487/RFC9254, July 2022, <https://www.rfc-editor.org/info/rfc9254>.

### 12.2. Informative References


**[I-D.ietf-netconf-distributed-notif]** Zhou, T., Zheng, G., Voit, E., Graf, T., and P. Francois, "Subscription to Distributed Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-distributed-notif-05, 6 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-distributed-notif-05>.

**[I-D.ietf-netconf-notification-messages]** Voit, E., Jenkins, T., Birkholz, H., Bierman, A., and A. Clemm, "Notification Message Headers and Bundles", Work in Progress, Internet-Draft, draft-ietf-netconf-notification-messages-08, 17 November 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-notification-messages-08>.

**[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>.

**[RFC8040]** Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

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

**[RFC8641]** Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

**[RFC8799]** Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <https://www.rfc-editor.org/info/rfc8799>.

**[RFC9147]** Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.


## Appendix A. UDP-notif Examples

This non-normative section shows two examples of how the the "ietf-udp-notif-transport" YANG module can be used to configure a [RFC8639] based publisher to send notifications to a receiver and an example of a YANG Push notification message using UDP-notif transport protocol.

### A.1. Configuration for UDP-notif transport with DTLS disabled

This example shows how UDP-notif can be configured without DTLS encryption.

```
================ NOTE: '\' line wrapping per RFC 8792 ================

<?xml version='1.0' encoding='UTF-8'?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-\
notifications">
    <subscription>
      <id>6666</id>
      <stream-subtree-filter>some-subtree-filter</stream-subtree-fil\
ter>
      <stream>some-stream</stream>
      <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-not\
if-transport">unt:udp-notif</transport>
      <encoding>encode-json</encoding>
      <receivers>
        <receiver>
          <name>subscription-specific-receiver-def</name>
          <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:\
ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</rece\
iver-instance-ref>
        </receiver>
      </receivers>
      <periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <period>6000</period>
      </periodic>
    </subscription>
    <receiver-instances xmlns="urn:ietf:params:xml:ns:yang:ietf-subs\
cribed-notif-receivers">
      <receiver-instance>
        <name>global-udp-notif-receiver-def</name>
        <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-\
udp-notif-transport">
          <address>192.0.5.1</address>
          <port>12345</port>
          <enable-segmentation>false</enable-segmentation>
          <max-segment-size/>
        </udp-notif-receiver>
      </receiver-instance>
    </receiver-instances>
  </subscriptions>
</config>
```

### A.2. Configuration for UDP-notif transport with DTLS enabled

This example shows how UDP-notif can be configured with DTLS encryption.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<?xml version='1.0' encoding='UTF-8'?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-\
notifications">
    <subscription>
      <id>6666</id>
      <stream-subtree-filter>some-subtree-filter</stream-subtree-fil\
ter>
      <stream>some-stream</stream>
      <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-not\
if-transport">unt:udp-notif</transport>
      <encoding>encode-json</encoding>
      <receivers>
        <receiver>
          <name>subscription-specific-receiver-def</name>
          <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:\
ietf-subscribed-notif-receivers">global-udp-notif-receiver-dtls-def<\
/receiver-instance-ref>
        </receiver>
      </receivers>
      <periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <period>6000</period>
      </periodic>
    </subscription>
    <receiver-instances xmlns="urn:ietf:params:xml:ns:yang:ietf-subs\
cribed-notif-receivers">
      <receiver-instance>
        <name>global-udp-notif-receiver-dtls-def</name>
        <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-\
udp-notif-transport">
          <address>192.0.5.1</address>
          <port>12345</port>
          <enable-segmentation>false</enable-segmentation>
          <max-segment-size/>
          <dtls>
            <client-identity>
              <tls13-epsk>
                <local-definition>
                  <key-format>ct:octet-string-key-format</key-format>
                  <cleartext-key>BASE64VALUE=</cleartext-key>
                </local-definition>
                <external-identity>example_external_id</external-ide\
ntity>
                <hash>sha-256</hash>
                <context>example_context_string</context>
                <target-protocol>8443</target-protocol>
                <target-kdf>12345</target-kdf>
```

```
            </tls13-epsk>
          </client-identity>
          <server-authentication>
            <ca-certs>
              <local-definition>
                <certificate>
                  <name>Server Cert Issuer #1</name>
                  <cert-data>BASE64VALUE=</cert-data>
                </certificate>
                <certificate>
                  <name>Server Cert Issuer #2</name>
                  <cert-data>BASE64VALUE=</cert-data>
                </certificate>
              </local-definition>
            </ca-certs>
            <ee-certs>
              <local-definition>
                <certificate>
                  <name>My Application #1</name>
                  <cert-data>BASE64VALUE=</cert-data>
                </certificate>
                <certificate>
                  <name>My Application #2</name>
                  <cert-data>BASE64VALUE=</cert-data>
                </certificate>
              </local-definition>
            </ee-certs>
            <raw-public-keys>
              <local-definition>
                <public-key>
                  <name>corp-fw1</name>
                  <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                  <public-key>BASE64VALUE=</public-key>
                </public-key>
                <public-key>
                  <name>corp-fw2</name>
                  <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                  <public-key>BASE64VALUE=</public-key>
                </public-key>
              </local-definition>
            </raw-public-keys>
            <tls13-epsks/>
          </server-authentication>
          <keepalives>
            <test-peer-aliveness>
              <max-wait>30</max-wait>
              <max-attempts>3</max-attempts>
```

```
              </test-peer-aliveness>
            </keepalives>
          </dtls>
        </udp-notif-receiver>
      </receiver-instance>
    </receiver-instances>
  </subscriptions>
</config>
```

### A.3. YANG Push message with UDP-notif transport protocol

This example shows how UDP-notif is used as a transport protocol to send a "push-update" notification [RFC8641] encoded in JSON [RFC7951].

Assuming the publisher needs to send the JSON payload showed in Figure 6, the UDP-notif transport is encoded following the Figure 7. The UDP-notif message is then encapsulated in a UDP frame.

```
{
    "ietf-notification:notification": {
        "eventTime": "2023-02-10T08:00:11.22Z",
        "ietf-yang-push:push-update": {
            "id": 1011,
            "datastore-contents": {
                "ietf-interfaces:interfaces": [
                    {
                        "interface": {
                            "name": "eth0",
                            "oper-status": "up"
                        }
                    }
                ]
            }
        }
    }
}
```

Figure 6: JSON Payload to be sent

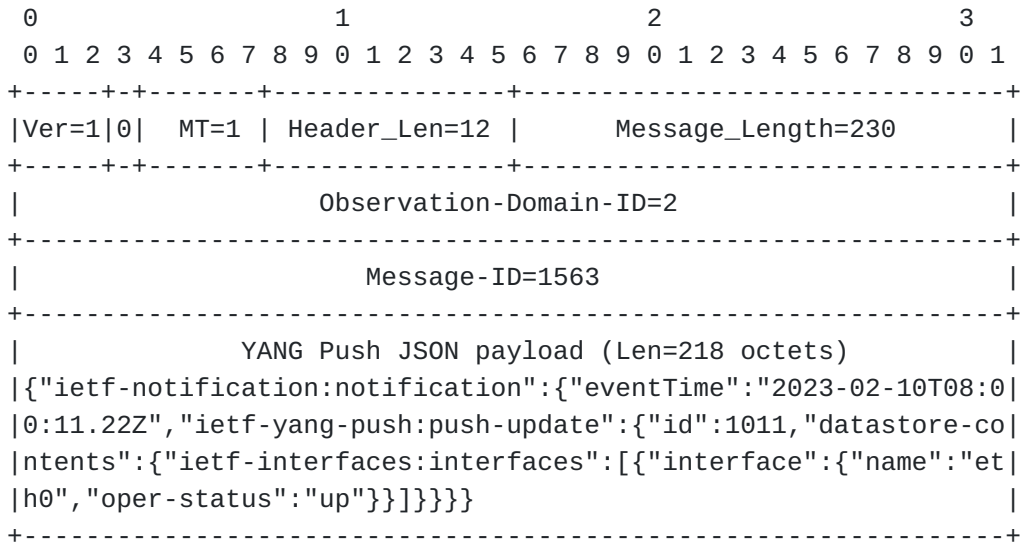


Figure 7: UDP-notif transport message

## Authors' Addresses


Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing
Jiangsu,
China

Email: zhengguangying@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: zhoutianran@huawei.com

Thomas Graf
Swisscom
Binzring 17
CH- Zuerich 8045
Switzerland

Email: thomas.graf@swisscom.com

Pierre Francois
INSA-Lyon
Lyon
France

Email: pierre.francois@insa-lyon.fr

Alex Huang Feng
INSA-Lyon
Lyon
France

Email: alex.huang-feng@insa-lyon.fr

Paolo Lucente
NTT
Siriusdreef 70-72
Hoofddorp, WT 2132
Netherlands

Email: paolo@ntt.net