

NETCONF  
Internet-Draft  
Intended status: Standards Track  
Expires: 17 April 2026

A. Huang-Feng  
P. Francois  
INSA-Lyon  
T. Zhou  
Huawei  
T. Graf  
Swisscom  
P. Lucente  
NTT  
14 October 2025

UDP-based Transport for Configured Subscriptions  
draft-ietf-netconf-udp-notif-23

Abstract

This document describes a UDP-based transport for YANG notifications to collect data from network nodes. A shim header is defined to facilitate the data streaming directly from a publishing process on a network device to telemetry receivers. Such a design enables higher frequency updates and less performance overhead on publisher and receiver processes compared to already established notification mechanisms. A YANG data model is also defined for management of the described UDP-based transport.

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 17 April 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. UDP-Based Transport . . . . .	5
3.1. Design Overview . . . . .	5
3.2. Format of the UDP-Notif Message Header . . . . .	6
3.3. Data Encoding . . . . .	8
4. Options . . . . .	8
4.1. Segmentation Option . . . . .	9
5. Applicability . . . . .	11
5.1. Congestion Control . . . . .	11
5.2. Message Size . . . . .	12
5.3. Reliability . . . . .	12
6. Secured layer for UDP-Notif . . . . .	12
6.1. Session Lifecycle . . . . .	13
6.1.1. DTLS Session Initiation . . . . .	13
6.1.2. Publish Data . . . . .	13
6.1.3. Session Termination . . . . .	14
6.1.4. DTLS Fragmentation . . . . .	14
7. A YANG Data Model for Management of UDP-Notif . . . . .	15
7.1. YANG Module for configuring UDP-Notif . . . . .	15
7.2. YANG Module . . . . .	17
8. IANA Considerations . . . . .	21
8.1. UDP-Notif Protocol Registry Group . . . . .	21
8.1.1. UDP-Notif Media Types Registry . . . . .	21
8.1.2. UDP-Notif Option Types Registry . . . . .	22
8.1.3. UDP-Notif Header Version Registry . . . . .	22
8.2. URI . . . . .	23
8.3. YANG Module Name . . . . .	23
9. Implementation Status . . . . .	23
9.1. Open Source Publisher . . . . .	23
9.2. Open Source Receiver Library . . . . .	23
9.3. Pmacct Data Collection . . . . .	24
9.4. Huawei VRP . . . . .	24
9.5. 6WIND VSR . . . . .	24
9.6. Cisco IOS XR . . . . .	24
10. Security Considerations . . . . .	24
11. Contributors . . . . .	25

12. Acknowledgements . . . . .	26
13. References . . . . .	26
13.1. Normative References . . . . .	26
13.2. Informative References . . . . .	28
Appendix A. UDP-Notif Examples . . . . .	30
A.1. Configuration for UDP-Notif transport with DTLS disabled . . . . .	30
A.2. Configuration for UDP-Notif transport with DTLS enabled . . . . .	32
A.3. YANG Push message with UDP-Notif transport protocol . . .	34
Authors' Addresses . . . . .	35

## 1. Introduction

The mechanism to support a subscription of a continuous and customized stream of updates from a YANG datastore [RFC8342] is defined in Subscribed Notifications [RFC8639] and YANG-Push [RFC8641].

Subscribed Notifications [RFC8639] separate 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] were defined 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 frequency with low overhead. This is important for network nodes that feature a distributed architecture with sparse resources on components specialized for packet forwarding. The currently available transports are TCP-based requiring the maintenance of connections, states and retransmissions, which is not necessary for high-frequency continuous notification content, typically published directly from network processors on line cards.

This document specifies a transport option for Configured Subscriptions as defined in Section 2.5 of [RFC8639] 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 UDP-based transport allows for such a distributed data publishing approach:

- \* Firstly, a UDP approach reduces the burden of maintaining a large pool of active TCP connections at the receiver, notably in cases where it collects data from network processors on line cards from a large number of network nodes.
- \* Secondly, as no connection state needs to be maintained, UDP encapsulation could be implemented by the hardware of the publisher, which further improves 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. It is designed to be used in cases where packet loss is not a concern, such as the collection of statistical metrics that are exported periodically. This transport can be configured via NETCONF [RFC6241] or RESTCONF [RFC8040].

This document describes the notification mechanism. It is intended to be used in conjunction with [RFC8639], extended by [I-D.ietf-netconf-distributed-notif]. Additionally, this document defines a YANG data model for management of the UDP-based transport. The YANG module specified in this document is compliant with Network Management Datastore Architecture (NMDA) [RFC8342].

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 mechanism. Section 6 describes a mechanism to secure the protocol in open networks. Finally, Section 7 defines a YANG data model for management of the mechanism described in this document.

## 2. Terminology

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

The following terms are used as defined in Subscribed Notifications [RFC8639]:

- \* Notification message
- \* Subscription
- \* Configured Subscription
- \* Subscriber
- \* Publisher
- \* Receiver

The following term is used as defined in [I-D.ietf-netconf-distributed-notif]:

- \* Message Publisher ID

This document defines the following term:

- \* Message ID: identifier of a message transported by the UDP-Notif protocol. More details are presented in Section 3.2.

### 3. UDP-Based Transport

This section specifies the UDP-Notif transport behavior. Section 3.1 describes the general design of the solution. Section 3.2 specifies the UDP-Notif message format and Section 3.3 describes the encoding of the message payload.

#### 3.1. Design Overview

As specified in Section 2.6 of Subscribed Notifications [RFC8639], the content of a YANG notification is encapsulated in a notification message, which is then encapsulated and carried using a transport protocol. Figure 1 illustrates the structure of a UDP-Notif message:

- \* The Message Header contains information that facilitates the message transmission before deserializing the notification message.
- \* The Notification Message is the encoded content that is transported by the publication stream. The common encoding methods are listed in Section 3.2. The structure of the notification message is defined in Section 2.6 of Subscribed Notifications [RFC8639].

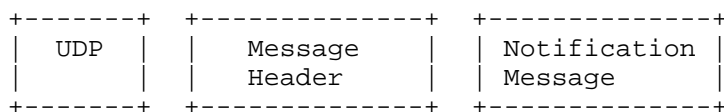


Figure 1: UDP-Notif Message Overview

When a publisher starts streaming UDP-Notif messages, the first message generated by the publisher **MUST** be a separate "subscription-started" notification to indicate to the receiver that the stream has started flowing. Then, the notifications can be sent immediately without delay. Subscription state notifications, defined in Section 2.7 of [RFC8639], **MUST** be encapsulated in separate notification messages.

Note that receivers collecting UDP-Notif messages may not be already up and running when the configuration of the subscription takes effect on a monitored network node.

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

The UDP-Notif message header contains information that facilitates the message transmission between the publisher and the receiver before deserializing the notification message. The data format is shown in Figure 2.

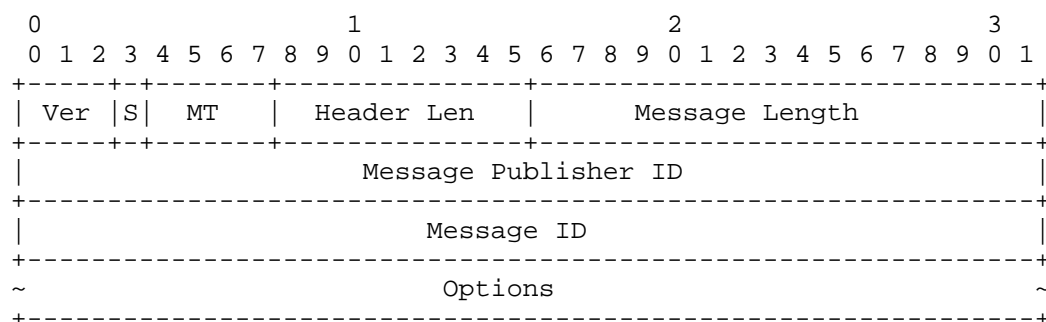


Figure 2: UDP-Notif Message Header Format

The Message Header contains the following field:

- \* Ver indicates the UDP-Notif protocol header version. The values are allocated by the IANA registry "UDP-Notif Header Version" (Section 8.1). The current header version number is 1.

- \* S-flag represents the space of media type specified in the MT field. When S-flag is not set, MT represents the standard media types as defined in the IANA registry "UDP-Notif media types" (Section 8.1). When S-flag is set, MT represents a private space to be freely used for non-standard encodings.
- \* MT is a 4-bit identifier that indicates the media type used for the notification message. When the S bit is not set, the following values apply:
  - 0: Reserved, MUST NOT be used.
  - 1: application/yang-data+json [RFC8040]
  - 2: application/yang-data+xml [RFC8040]
  - 3: application/yang-data+cbor [RFC9254]
- \* Header Len (8-bit) records the length of the message header in octets, including both the fixed header and the options.
- \* Message Length (16-bit) records the total length of the UDP-Notif message within one UDP datagram, measured in octets, including the message header. When the notification message is segmented using the Segmentation Options defined in Section 4.1, the Message Length is the total length of the current UDP-Notif segment, not the length of the entire notification message.
- \* Message Publisher ID is a 32-bit identifier defined in [I-D.ietf-netconf-distributed-notif]. This identifier is locally unique to the publisher node. It identifies the software process generating the stream of UDP-Notif messages and allow the disambiguation of an information source. Message unicity is obtained from the conjunction of the Message Publisher ID and the Message ID field. If Message Publisher ID unicity is not preserved through the collection domain, the source IP address of the UDP datagram MUST be used in addition to the Message Publisher ID to identify the information source. If a transport layer relay is used, Message Publisher ID unicity must be preserved through the collection domain.
- \* The Message ID is increased monotonically by the publisher of UDP-Notif messages and MUST start at 1 with the first message. A publisher MUST use different Message IDs for different messages generated with the same Message Publisher ID. Note that the main purpose of the Message ID is to reconstruct messages which are segmented using the segmentation option described in Section 4.1. The Message ID values SHOULD be incremented by one for successive

messages originated with the same Message Publisher ID, so that message loss can be detected at data collection. When the last value ( $2^{32}-1$ ) of Message ID has been reached, the Message ID wraps around and restarts at 0.

- \* Options are 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. Options are described in Section 4.

All the binary fields MUST be encoded in network byte order (big endian).

### 3.3. Data Encoding

UDP-Notif message data can be encoded in XML, JSON or CBOR format. 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. The new encoding can be registered in the IANA registry "UDP-Notif media types" following the procedure defined in Section 8.1.

Subscribed Notifications [RFC8639] states that a transport MUST identify a default encoding. However, as per [Errata-6211], Subscribed Notifications does not require to define a default encoding.

Private encodings can be used by enabling the S-flag of the header. When the S-flag is set, the value of the MT field is left to be defined and agreed upon by the users of the private encoding. The MT field allows for 16 private encodings when S-flag is set.

The encoding of a message data is configured on a subscription basis and each subscription reference a receiver instance. Publishers MUST NOT be configured to send notification messages with more than one encoding to the same receivers.

## 4. Options

All the options are defined with the format shown in Figure 3.

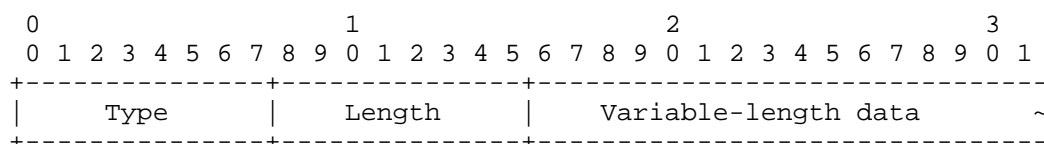




Figure 3: Generic Option Format

- \* Type: 1-octet describing the option type. The values of the Type field are allocated by the IANA registry "UDP-Notif options types" (Section 8.1).
- \* 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 data.

When more than one option are used in a UDP-Notif header, the segmentation option defined in Section 4.1 MUST be placed first, if present. Placing the segmentation option first can simplify some implementations for both the publisher and the receiver, notably those assuming a fixed location for the segmentation option. Segmented messages where the segmentation option is not the first option MAY be discarded by the receiver.

4.1. Segmentation Option

The UDP payload length is limited to 65527 bytes (65535 - 8 bytes). 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 [BCP230].

Implementations MUST provide a configurable parameter to control the maximum size of a UDP-Notif segment. This parameter is defined as "max-segment-size" in the YANG module specified in Section 7.1.

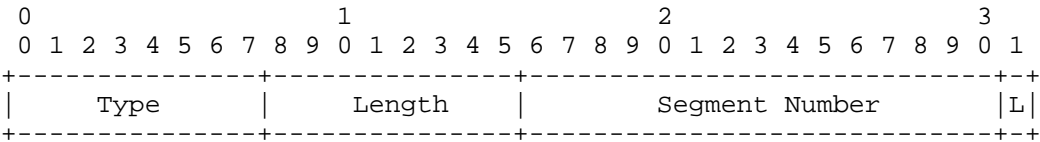


Figure 4: Segmentation Option Format

The Segmentation Option (Figure 4) is included when the message content is segmented into multiple segments. Different segments of one message share the same Message ID. The fields of this option are as follows:

- \* Type: indicates a Segmentation Option. The value is 1 for this option.
- \* Length: indicates the length of this option, in octets. It MUST be set to 4 octets.
- \* 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. The segment number cannot wrap around.
- \* L: indicates 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.

Implementations MUST NOT rely on IP fragmentation to carry large messages. Implementations MUST either restrict the size of individual messages to a value that will not lead to IP fragmentation as per Section 5.2, or support the segmentation option. In the latter case, the parameter "max-segment-size" MUST be set so that the size of a UDP-Notif segment and the size of the IP layer together do not exceed the MTU of the egress interface.

When a message has multiple options and is segmented, all the options MUST be present on the first segment. The rest of segmented messages MAY include all the options. The segmentation option MUST be placed first in all segments.

The receiver SHOULD support the reception of unordered segments. The implementation of the receiver SHOULD provide an option to discard the received segments if, after some time, one of the segments is still missing and the reassembly of the message is not possible. No retransmission of lost segments are expected from the publisher. If the receiver collects a segment more than once, the implementation SHOULD drop the duplicate segment.

To reassemble segmented UDP-Notif messages, the receiver should first identify UDP-Notif segments belonging to the same message by using the combination of the Message Publisher ID and Message ID. The receiver SHOULD wait for all the segments before starting the reassembly process. Once all the segments are collected, the receiver should create a new UDP-Notif header with the same Ver, S-flag, MT, Message Publisher ID and Message ID values. When UDP-Notif options other than the segmentation option are present in the first segment, these options need to be appended to the newly created UDP-Notif header. To reconstruct the original notification message, the receiver must concatenate the notification message of each UDP-

Notif segments in an ascending order based on the Segment Number. The new concatenated notification message becomes the notification message of the newly created UDP-Notif message. The Header Length and Message Length are then updated accordingly.

## 5. Applicability

This section provides an applicability for the UDP-Notif mechanism, following the recommendations of [RFC8085].

The 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 SHOULD thus follow the recommendations in place for such specific applications. We discuss recommendations on congestion control in Section 5.1, message size guidelines in Section 5.2 and reliability considerations in Section 5.3.

The main use case of the UDP-Notif mechanism is the collection of statistical metrics for accounting purposes, where potential loss is not a concern, but should however be reported (such as IPFIX Flow Records exported with UDP [RFC7011]). Such metrics are typically exported in a periodical subscription as described in Section 3.1 of [RFC8641].

### 5.1. Congestion Control

The above application falls into the category of applications performing transfer of large amounts of data. It is expected that the operator using the solution configures dedicated class of services 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 UDP-Notif 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 SHOULD NOT push unbounded volumes 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 publisher to use a lower streaming rate. The interaction to control the streaming rate on the publisher 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.

It is RECOMMENDED that the size of a Notification Message is small and segmentation does not result in segmenting the message into too many segments to avoid dropping the entire message when there is a lost segment.

A receiver collecting segmented UDP-Notif messages SHOULD have a configurable parameter to discard segments when they exceed a certain amount of segments. The generation of too many segments by a publisher can be used as an abuse to require computation resources for reassembling large messages at the receiver.

## 5.3. Reliability

A receiver implementation SHOULD discard 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 unsecured networks, which are not authenticated and encrypted on layers below transport, UDP-Notif messages MUST be encrypted. This section presents a mechanism using DTLS [RFC6347][RFC9147] to secure UDP-Notif protocol. In addition to providing encryption, DTLS also ensures authentication and integrity protection, preventing attacks such as the injection of malicious packets.

Implementations using DTLS to secure UDP-Notif messages MUST support DTLS 1.2 [RFC6347] or later, and SHOULD support DTLS 1.3 [RFC9147]. No DTLS extensions are defined in this document.

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 configured 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 number, 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

This section describes the lifecycle of UDP-Notif messages when they are encrypted using DTLS.

### 6.1.1. DTLS Session Initiation

The publisher initiates a DTLS connection by sending a DTLS ClientHello to the Receiver. Implementations MAY disable the denial of service countermeasures defined by DTLS 1.2 and DTLS 1.3 if a given deployment can ensure that DoS attacks are not a concern.

In DTLS 1.3 when the denial of service countermeasures are implemented, the Receiver responds with a DTLS HelloRetryRequest containing a stateless cookie. The publisher sends then a second DTLS ClientHello message containing the received cookie. Details can be found in Section 5.1 of [RFC9147].

When DTLS is implemented, the publisher MUST NOT send any UDP-Notif messages before the DTLS handshake has successfully completed. Implementations MUST NOT use the early data mechanism (also known as 0-RTT data) defined in DTLS 1.3 [RFC9147].

Implementations MUST follow recommendations defined by [BCP195]. If other cipher suites than the ones recommended by [BCP195] are used, then implementations MUST NOT negotiate a cipher suite that employs NULL integrity or authentication algorithms.

Where confidentiality protection with DTLS is required, implementations must negotiate a cipher suite that employs a non-NULl encryption algorithm.

### 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.

#### 6.1.4. DTLS Fragmentation

DTLS 1.2 [RFC6347] and DTLS 1.3 [RFC9147] states that DTLS message may be fragmented into multiple DTLS records. A DTLS message carrying a UDP-Notif message SHOULD fit within a single datagram to avoid DTLS fragmentation. Implementations SHOULD account for DTLS overhead when determining the maximum UDP-Notif notification message size.

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

### 7.1. YANG Module for configuring UDP-Notif

The YANG model described in Section 7.2 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.

The source address of the UDP-Notif message can be configured using the "source-address" leaf at the subscription level as defined in Section 2.5 of [RFC8639] or by setting the leaf "local-address" using the "ietf-udp-notif-transport" YANG module. When both are configured, the UDP-Notif message MUST use the address configured in the "local-address" leaf defined in the "ietf-udp-notif-transport" YANG module.

The model defines the following YANG tree [RFC8340]:

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

```
augment /sn:subscriptions/snr:receiver-instances
  /snr:receiver-instance/snr:transport-type:
  +--:(udp-notif)
  +--rw udp-notif-receiver
    +--rw remote-address          inet:host
    +--rw remote-port             inet:port-number
    +--rw local-address?          inet:ip-address
    |   {local-binding}?
    +--rw local-port?             inet:port-number
    |   {local-binding}?
    +--rw dtls! {dtls}?
    |   +--rw client-identity!
    |   |   +--rw (auth-type)
    |   |   |   +--:(certificate) {client-ident-x509-cert}?
    |   |   |   |   ...
    |   |   |   +--:(raw-public-key)
    |   |   |   |   {client-ident-raw-public-key}?
    |   |   |   |   ...
    |   |   |   +--:(tls12-psk) {client-ident-tls12-psk}?
    |   |   |   |   ...
    |   |   |   +--:(tls13-epsk) {client-ident-tls13-epsk}?
    |   |   |   |   ...
    |   +--rw server-authentication
    |   |   +--rw ca-certs! {server-auth-x509-cert}?
    |   |   |   +--rw (inline-or-truststore)
    |   |   |   |   ...
    |   |   +--rw ee-certs! {server-auth-x509-cert}?
    |   |   |   +--rw (inline-or-truststore)
    |   |   |   |   ...
    |   |   +--rw raw-public-keys! {server-auth-raw-public-key}?
    |   |   |   +--rw (inline-or-truststore)
    |   |   |   |   ...
    |   |   +--rw tls12-psks?      empty
    |   |   |   {server-auth-tls12-psk}?
    |   |   +--rw tls13-epsks?    empty
    |   |   |   {server-auth-tls13-epsk}?
    |   +--rw hello-params {tlscmn:hello-params}?
    |   +--rw tls-versions
    |   |   +--rw min?      identityref
    |   |   +--rw max?      identityref
    |   +--rw cipher-suites
    |   |   +--rw cipher-suite*
    |   |   |   tlscsa:tls-cipher-suite-algorithm
    +--rw enable-segmentation?  boolean
    +--rw max-segment-size?     uint16
```



## 7.2. 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" defines the necessary parameters to configure the transport defined in this document using the generic "udp-client" grouping from the "ietf-udp-client" module [I-D.ietf-netconf-udp-client-server] and the "tls-client-grouping" defined in the "ietf-tls-client" module [RFC9645]. It uses data types defined in [RFC6991].

```
<CODE BEGINS> file "ietf-udp-notif-transport@2025-06-04.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
      "draft-ietf-netconf-https-notif: An HTTPS-based Transport
       for Configured Subscriptions";
  }
  import ietf-udp-client {
    prefix udpc;
    reference
      "draft-ietf-netconf-udp-client-server: YANG Grouping for
       UDP Clients and UDP Servers";
  }
  import ietf-tls-client {
    prefix tlsc;
    reference
      "RFC 9645: 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:   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 a model for configuring UDP-Notif as a transport for configured subscriptions [RFC8639].

Copyright (c) 2025 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 (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.

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 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.";

```
revision 2025-06-04 {
  description
    "Initial revision";
  reference
    "RFC XXXX: UDP-based Transport for Configured Subscriptions";
}
```

```
/*
 * FEATURES
 */
```

```
feature encode-cbor {
  description
    "Indicates that CBOR encoding of notification
    messages is supported.";
```

```
    reference
      "RFC 9254: CBOR Encoding of Data Modeled with YANG";
  }

  feature dtls {
    description
      "Indicates that DTLS encryption of UDP
      packets is supported. UDP-Notif mandates that, in
      unsecured networks, DTLS 1.2 or later MUST be supported,
      and DTLS 1.3 SHOULD be supported.";
    reference
      "RFC6347: Datagram Transport Layer Security Version 1.2,
      RFC 9147: The Datagram Transport Layer Security (DTLS)
      Protocol Version 1.3";
  }

  /*
   * IDENTITIES
   */

  identity udp-notif {
    base sn:transport;
    base sn:configurable-encoding;
    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.";
    reference
      "RFC 9254: CBOR Encoding of Data Modeled with YANG";
  }

  identity unsupported-max-segment-size {
    base sn:establish-subscription-error;
    base sn:modify-subscription-error;
    description
      "Error triggered when the specified value 'max-segment-size'
      is not supported by the publisher. An implementation may
      only support a subset of the uint16.";
    reference
      "RFC XXXX: UDP-based Transport for Configured Subscriptions";
  }

  grouping udp-notif-receiver {
```

```
description
  "Provides a reusable identification of a UDP-Notif target
  receiver.";
uses udpc:udp-client {
  refine "remote-port" {
    mandatory true;
  }
}
container dtls {
  if-feature "dtls";
  presence "dtls";
  uses tlsc:tls-client-grouping {
    // Remove keep-alives for DTLS
    refine "keepalives" {
      if-feature "not tlsc:tls-client-keepalives";
    }
  }
  description
    "Container for configuring DTLS parameters.";
}
leaf enable-segmentation {
  type boolean;
  default "true";
  description
    "When disabled, the publisher will not segment UDP-Notif
    messages. This may cause IP-layer fragmentation when
    messages are larger than the MTU. IP fragmentation is
    discouraged (RFC 8085, RFC 8900) and generally unsafe.
    Disabling is not recommended.";
}
leaf max-segment-size {
  type uint16;
  description
    "UDP-Notif provides a configurable max-segment-size to
    control the size of each segment (UDP-Notif header, with
    options, included).
    The publisher may trigger an 'unsupported-max-segment-size'
    error if the publisher does not support the configured
    value.";
}
}

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;
    }
}
description
    "Augments the transport-type choice to include the 'udp-notif'
    transport.";
}
}
<CODE ENDS>

```

## 8. IANA Considerations

This document defines new registries under a new registry group entitled "UDP-Notif Protocol", and updates the IETF XML and YANG module registries.

### 8.1. UDP-Notif Protocol Registry Group

This document requests IANA to create a new registry group called "UDP-Notif protocol". Under this registry group, three registries are to be created as described in the following sections.

#### 8.1.1. UDP-Notif Media Types Registry

All UDP-Notif messages contain a 4-bit media type identifier, for which IANA is to create and maintain a new registry entitled "UDP-Notif Media Types" under the registry group "UDP-Notif Protocol". This document defines the following media type values:

Value	Description	Reference
0	Reserved	RFC-to-be
1	media type application/yang-data+json	RFC8040
2	media type application/yang-data+xml	RFC8040
3	media type application/yang-data+cbor	RFC9254

Table 1: Initial UDP-Notif Media Types Registry

Future assignments are to be made using the Standards Action process defined in Section 4.9 of [RFC8126]. Assignments consist of the value, a short description of the media type and the document reference (e.g., RFC number).

### 8.1.2. UDP-Notif Option Types Registry

UDP-Notif uses an 8-bit option type (see Section 4), for which IANA is to create and maintain a new registry entitled "UDP-Notif Option Types" under the registry group "UDP-Notif Protocol". This document defines the following option type values:

Value	Description	Reference
0	Reserved	RFC-to-be
1	Segmentation Option	RFC-to-be

Table 2: Initial UDP-Notif Option Types Registry

Future assignments are to be made using the Standards Action process defined in Section 4.9 of [RFC8126]. Assignments consist of the value, a short description of the option and the document reference (e.g., RFC number).

### 8.1.3. UDP-Notif Header Version Registry

UDP-Notif header uses a 3-bit header version, for which IANA is to create and maintain a new registry entitled "UDP-Notif Header Version" under the registry group "UDP-Notif Protocol". This document defines the following header version values:

Value	Description	Reference
0	UDP based Publication Channel for Streaming Telemetry	draft-ietf-netconf-udp-pub-channel-05
1	UDP-based Transport for Configured Subscriptions	RFC-to-be

Table 3: Initial UDP-Notif Header Version Registry

Note: There is an older specification of this transport protocol defined in [I-D.ietf-netconf-udp-pub-channel] that was deployed in some networks. To enable differentiating both protocols, different version numbers are used. The current specification replaces [I-D.ietf-netconf-udp-pub-channel] and uses 1 as its version, while the header defined in [I-D.ietf-netconf-udp-pub-channel] uses 0.

Future assignments are to be made using the Standards Action process defined in Section 4.9 of [RFC8126]. Assignments consist of the value, a description of the header version and the document reference (e.g., RFC number).

## 8.2. URI

IANA is also requested to register the following URI in the "ns" registry within the "IETF XML Registry" group [RFC3688]:

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.

## 8.3. YANG Module Name

IANA is requested to register the following YANG module in the "YANG Module Names" registry [RFC6020] within the "YANG Parameters" registry group.

Name: ietf-udp-notif-transport  
Maintained by IANA: N  
Namespace: urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport  
Prefix: unt  
Reference: RFC-to-be

## 9. Implementation Status

Note to the RFC-Editor: Please remove this section before publishing.

### 9.1. Open Source Publisher

INSA Lyon implemented this document for a YANG Push publisher in an example implementation.

The open source code can be obtained here: [INSA-Lyon-Publisher].

### 9.2. Open Source Receiver Library

INSA Lyon implemented this document for a YANG Push receiver as a library.

The open source code can be obtained here: [INSA-Lyon-Receiver].

### 9.3. Pmacct Data Collection

The open source YANG push receiver library [INSA-Lyon-Receiver] has been integrated into the Pmacct open source Network Telemetry data collection [Paolo-Lucente-Pmacct].

### 9.4. Huawei VRP

Huawei implemented this document for a YANG Push publisher in their VRP platform.

### 9.5. 6WIND VSR

6WIND implemented this document for a YANG Push publisher in their VSR platform.

### 9.6. Cisco IOS XR

Cisco implemented this document for a YANG Push publisher in their IOS XR platform.

## 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, for networks that are not secured, a secure transport providing confidentiality, integrity protection, authentication, and replay protection MUST be implemented. A specification of UDP-Notif using DTLS 1.3 as its encryption layer is presented in Section 6.

The following text uses the template described in Section 3.7 of [I-D.ietf-netmod-rfc8407bis].

The "ietf-udp-notif-transport" YANG module defines a data model that is designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. These protocols have to use a secure transport layer (e.g., SSH [RFC6242], TLS [RFC8446], and QUIC [RFC9000]) and have to use mutual authentication.



The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

- \* The data nodes "remote-address", "remote-port", "local-address", and "local-port" in the "ietf-udp-notif-transport" module specify transport parameters for the recipient of UDP-Notif messages. Unauthorized modification of these transport parameters could redirect notifications to unintended recipients.

This YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Refer to the Security Considerations of [I-D.ietf-netconf-udp-client-server] and [RFC9645] for information as to which nodes may be considered sensitive or vulnerable in network environments.

## 11. Contributors

Guangying Zheng  
Huawei  
101 Yu-Hua-Tai Software Road  
Nanjing  
Jiangsu,  
China  
Email: zhengguangying@huawei.com

Yunan Gu  
Huawei  
Beijing  
China  
Email: guyunan@huawei.com

## 12. Acknowledgements

The authors of this documents would like to thank Lucas Aubard, Alexander Clemm, Benoit Claise, Ebben Aries, Eric Voit, Huiyang Yang, Kent Watsen, Mahesh Jethanandani, Marco Tollini, Hannes Tschofenig, Michael Tuxen, Rob Wilton, Sean Turner, Stephane Frenot, Timothy Carey, Tim Jenkins, Tom Petch, Joseph Touch, Andy Bierman, Carsten Bormann, Mohamed Boucadair, Weiqiang Cheng, Giuseppe Fioccola, Camilo Cardona, Qiufang Ma, James Cumming and Qin Wu for their constructive suggestions for improving this document.

## 13. References

### 13.1. Normative References

- [I-D.ietf-netconf-distributed-notif]  
Zhou, T., Zheng, G., Voit, E., Graf, T., and P. Francois,  
"Subscription to Notifications in a Distributed  
Architecture", Work in Progress, Internet-Draft, draft-  
ietf-netconf-distributed-notif-15, 4 July 2025,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-distributed-notif-15>>.
- [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-15, 1 February 2024,  
<<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-https-notif-15>>.
- [I-D.ietf-netconf-udp-client-server]  
Feng, A. H., Francois, P., and K. Watsen, "YANG Groupings  
for UDP Clients and UDP Servers", Work in Progress,  
Internet-Draft, draft-ietf-netconf-udp-client-server-08,  
10 October 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-udp-client-server-08>>.
- [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>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [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>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [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>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC9645] Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", RFC 9645, DOI 10.17487/RFC9645, October 2024, <<https://www.rfc-editor.org/info/rfc9645>>.

### 13.2. Informative References

- [BCP195] Best Current Practice 195, <<https://www.rfc-editor.org/info/bcp195>>. At the time of writing, this BCP comprises the following:
- Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <<https://www.rfc-editor.org/info/rfc8996>>.
- Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <<https://www.rfc-editor.org/info/rfc9325>>.
- [BCP230] Best Current Practice 230, <<https://www.rfc-editor.org/info/bcp230>>. At the time of writing, this BCP comprises the following:

Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", BCP 230, RFC 8900, DOI 10.17487/RFC8900, September 2020, <<https://www.rfc-editor.org/info/rfc8900>>.

[Errata-6211]

Watsen, Kent., "Errata 6211", 2024, <<https://www.rfc-editor.org/errata/eid6211>>.

[I-D.ietf-netconf-udp-pub-channel]

Zheng, G., Zhou, T., and A. Clemm, "UDP based Publication Channel for Streaming Telemetry", Work in Progress, Internet-Draft, draft-ietf-netconf-udp-pub-channel-05, 11 March 2019, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-udp-pub-channel-05>>.

[I-D.ietf-netmod-rfc8407bis]

Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-28, 5 June 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28>>.

[INSA-Lyon-Publisher]

"INSA Lyon, YANG Push publisher example implementation", <<https://github.com/network-analytics/udp-notif-scapy>>.

[INSA-Lyon-Receiver]

"INSA Lyon, YANG Push receiver library implementation", <<https://github.com/network-analytics/udp-notif-c-collector>>.

[Paolo-Lucente-Pmacct]

"Paolo Lucente, Pmacct open source Network Telemetry Data Collection", <<https://github.com/pmacct/pmacct>>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.

- [RFC7011] Claise, B., Ed., Trammell, B., Ed., and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information", STD 77, RFC 7011, DOI 10.17487/RFC7011, September 2013, <<https://www.rfc-editor.org/info/rfc7011>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.

## 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. It illustrates the definition of two receivers, one uses an IPv4 as its destination address and another uses IPv6. The IPv4 receiver is bound to the subscription.

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

```
<?xml version='1.0' encoding='UTF-8'?>
<subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
  <subscription>
    <id>6666</id>
    <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">ds:operational</datastore>
    <datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
      xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">/if:interfaces/interface</datastore-xpath-filter>
    <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif">unt:udp-notif</transport>
    <encoding>encode-json</encoding>
    <receivers>
      <receiver>
        <name>subscription-specific-receiver</name>
        <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">ipv4-udp-notif-receiver</receiver-instance-ref>
        <state>active</state>
      </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-subscribed-notif-receivers">
    <receiver-instance>
      <name>ipv4-udp-notif-receiver</name>
      <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">
        <remote-address>192.0.2.1</remote-address>
        <remote-port>12345</remote-port>
        <enable-segmentation>true</enable-segmentation>
        <max-segment-size>9000</max-segment-size>
      </udp-notif-receiver>
    </receiver-instance>
    <receiver-instance>
      <name>ipv6-udp-notif-receiver</name>
      <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">
        <remote-address>2001:db8:abcd:12::1</remote-address>
        <remote-port>12345</remote-port>
```

```

        <enable-segmentation>true</enable-segmentation>
        <max-segment-size>9000</max-segment-size>
    </udp-notif-receiver>
</receiver-instance>
</receiver-instances>
</subscriptions>

```

## 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'?>
<subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
  <subscription>
    <id>6666</id>
    <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">ds:operational</datastore>
    <datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
      xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">/if:interfaces/interface</datastore-xpath-filter>
    <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif">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">udp-notif-receiver-dtls</receiver-instance-ref>
        <state>active</state>
      </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-subscribed-notif-receivers">
    <receiver-instance>
      <name>udp-notif-receiver-dtls</name>
      <udp-notif-receiver xmlns="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">

```



```
<remote-address>2001:db8:abcd:12::1</remote-address>
<remote-port>12345</remote-port>
<dtls>
  <client-identity>
    <tls13-epsk>
      <inline-definition>
        <key-format xmlns:ct="urn:ietf:params:xml:ns:yang:ie\
tf-crypto-types">ct:octet-string-key-format</key-format>
        <cleartext-symmetric-key>BASE64VALUE=</cleartext-sym\
metric-key>
      </inline-definition>
      <external-identity>example_external_id</external-ident\
ity>
      <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>
      <inline-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>
      </inline-definition>
    </ca-certs>
    <ee-certs>
      <inline-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>
      </inline-definition>
    </ee-certs>
    <raw-public-keys>
      <inline-definition>
        <public-key>
          <name>corp-fw1</name>
```

```

        <public-key-format xmlns:ct="urn:ietf:params:xml:n\
s:yang:ietf-crypto-types">ct:subject-public-key-info-format</public-\
key-format>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
    <public-key>
      <name>corp-fw2</name>
      <public-key-format xmlns:ct="urn:ietf:params:xml:n\
s:yang:ietf-crypto-types">ct:subject-public-key-info-format</public-\
key-format>
        <public-key>BASE64VALUE=</public-key>
      </public-key>
    </inline-definition>
  </raw-public-keys>
  <tls13-epsks/>
</server-authentication>
</dtls>
<enable-segmentation>true</enable-segmentation>
<max-segment-size>9000</max-segment-size>
</udp-notif-receiver>
</receiver-instance>
</receiver-instances>
</subscriptions>

```

### 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 5, the UDP-Notif transport is encoded following the Figure 6. The UDP-Notif message is then encapsulated in a UDP datagram.

```

{
  "ietf-notification:notification": {
    "eventTime": "2024-02-10T08:00:11.22Z",
    "ietf-yang-push:push-update": {
      "id": 1011,
      "datastore-contents": {
        "ietf-interfaces:interfaces": [
          {
            "interface": {
              "name": "eth0",
              "oper-status": "up"
            }
          }
        ]
      }
    }
  }
}

```

Figure 5: JSON Payload to be sent

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Ver=1 0										MT=1										Header_Len=12										Message_Length=230									
Message Publisher ID=2																																							
Message ID=1563																																							
YANG Push JSON payload (Len=218 octets)																																							
{"ietf-notification:notification":{"eventTime":"2024-02-10T08:00:11.22Z","ietf-yang-push:push-update":{"id":1011,"datastore-contents":{"ietf-interfaces:interfaces":[{"interface":{"name":"eth0","oper-status":"up"}}]}}}}																																							

Figure 6: UDP-Notif transport message

## Authors' Addresses

Alex Huang Feng  
 INSA-Lyon  
 Lyon  
 France  
 Email: alex.huang-feng@insa-lyon.fr

Pierre Francois  
INSA-Lyon  
Lyon  
France  
Email: pierre.francois@insa-lyon.fr

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

Paolo Lucente  
NTT  
Siriusdreef 70-72  
Hoofddorp, WT 2132  
Netherlands  
Email: paolo@ntt.net