NETCONF
Internet-Draft
Intended status: Standards Track
Expires: 18 July 2025

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

UDP-based Transport for Configured YANG Subscriptions
draft-ietf-netconf-udp-notif-18

Commenté [MB1]: To demux which subscriptions we are talking about

Abstract

This document describes a UDP-based ~~protocol~~ transport for YANG notifications to collect data from network nodes. A shim header is ~~proposed~~ defined to facilitate the data streaming directly from the publishing process on network processor of line cards to receivers. ~~The objective is to~~Such a design ~~provide a lightweight approach to~~ enables higher frequency and less performance impact on publisher and receiver processes compared to already established notification mechanisms.

Commenté [MB2]: Hope this was formally assessed.

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 18 July 2025.

Table of Contents

1. Introduction

The mechanism to support a subscription of a continuous and customized stream of updates from a YANG datastore [RFC8342] is defined in [RFC8639] and [RFC8641] and is abbreviated as Sub-Notif. Requirements for Subscription to YANG Datastores are defined in [RFC7923].

The [RFC8342] 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~~were ~~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 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 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 pool 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 network nodes.

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

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

a mis en forme : Surlignage

a mis en forme : Surlignage

a mis en forme : Surlignage

Commenté [MB3]: Seems to be tautologique :-)

a mis en forme : Surlignage

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.

X. Terms

Commenté [MB4]: Consider introducing the terms used in the doc.

2. Configured Subscription to UDP-Notif

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

As specified in Sub-Notif [REF], 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.

Commenté [MB5]: Add explicit section of the doc

Commenté [MB6]: Transport?

Commenté [MB7]: Sub-Notif does not cover UDP. Please reword.

Note that receivers ~~MAY NOT~~may not be already up and running when the configuration of the subscription takes effect on ~~the~~ a monitored network node. The first message MUST be a separate "subscription-started" notification to indicate to the ~~Receiver~~ receiver that the stream has started flowing. Then, the notifications can be sent immediately without delay. Subscription state change notifications~~All the subscription state notifications~~, ~~as~~ defined in Section 2.7 of [RFC8639], MUST be encapsulated in separate notification messages.

Commenté [MB8]: From where to?

Commenté [MB9]: What if reordering or loss happens for the -started message?

Note also that the receiver nodes can be different from the nodes managing the subscription. Therefore, publishers ~~MAY NOT~~may not be aware of the capabilities supported by the receivers.

Commenté [MB10]: This is not specific to this spec.

3. UDP-Based Transport

~~In t~~This section~~, we specify~~ specifies the UDP-Notif t~~T~~ransport 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 Sub-Notif, the YANG data is encapsulated in a NETCONF/RESTCONF notification message, which is then encapsulated and

Commenté [MB11]: Please indicate the exact section of that spec

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

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

* The n~~N~~otification ~~Message~~ 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~~ notification ~~Message~~ message is defined in Section 2.6 of [RFC8639] and a YANG model ~~has been~~is proposed in [I-D.ahuang-netconf-notif-yang]. [I-D.ietf-netconf-notification-messages] ~~proposes~~ specifies a structure to send bundled notifications in a single message.

Commenté [MB12]: Consistency with 8639. Please update through the doc

Commenté [MB13]: This was replaced by draft-netana-netconf-notif-envelope-02 - Extensible YANG Model for YANG-Push Notifications

Commenté [MB14]: De we need to include this?

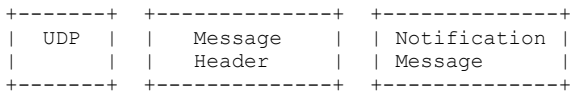

Figure 1: UDP-Notif Message Overview

3.2. Format of the UDP-Notif Message Header

The UDP-Notif m~~M~~essage ~~Header~~ header contains information that facilitates the message transmission before deserializing the notification message. The data format is shown in Figure 2.

Commenté [MB15]: Where?

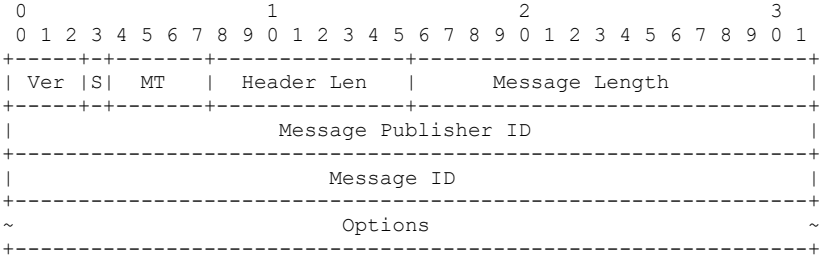

Figure 2: UDP-Notif Message Header Format

The Message Header contains the following fields:

* "Ver" indicates the UDP-Notif protocol header version. The values are allocated by the IANA registry "UDP-Notif header version". The current header version number is 1.

Commenté [MB16]: Add a pointer to the IANA section

* S-flag represents the space of media type specified in the MT field.

When S-flag 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~~non-standard encodings. When S is set, the Private Encoding Option defined in Section 4.2 SHOULD be present in the UDP-Notif message header.

* MT is a ~~4 bit~~4-bit identifier ~~to~~ that indicates the media type used for the ~~Notification~~ notification ~~Message~~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 (8-bit) ~~is~~ records the length of the message header in octets, including both the fixed header and the options.

* Message Length (16-bit) ~~is~~ records the total length of the UDP-Notif message within one UDP datagram, measured in octets, including the message header. When the n~~N~~otification ~~Message~~ message is segmented using the Segmentation Options defined in Section 4.1, the Message Length is the total length of the current, ~~segmented~~ UDP-Notif message segment, not the length of the entire ~~Notification~~ notification message.

* Message Publisher ID is a 32-bit identifier defined in [I-D.ietf-netconf-distributed-notif]. This identifier is unique to the publisher node and identifies the publishing process of the node to allow the disambiguation of an information source.

  Message unicity is obtained from the conjunction of the Message Publisher ID and the Message ID field ~~described below~~. If Message Publisher ID unicity is not preserved through the collection domain, the source IP address of the UDP datagram SHOULD 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 generated continuously by the publisher of UDP-Notif messages. A publisher MUST use different Message IDs ~~values~~ 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~~ Section 4.1. The Message ID values SHOULD be incremented by one for ~~each~~ successive ~~message~~messages originated with the same Message Publisher ID, so that message

Commenté [MB17]: That is, this bit will be set even if future types are registered?

Commenté [MB18]: How to decode if it is not present?

Commenté [MB19]: Redundant with 4-bit

Commenté [MB20]: MUST NOT be used. No?

Commenté [MB21]: What is the unicity scope? Who manages that? How conflicts are avoided? Is that problematic?

Commenté [MB22]: The distributed-id says the following: «"Identifies the software process which publishes the message (e.g., processor 1 on line card 1). This field is used to notify the receiver which publisher process published which message."; »

There is a disconnect between that description and the interpretation provided here.

Commenté [MB23]: What about also considering the transport coordinates?

Commenté [MB24]: I have the answer to my previous comment.

I suggest you simplify here and merge the three dimensions. No need for the normative language though.

Commenté [MB25]: How?

Commenté [MB26]: Is there any value in randomly selecting the initial ID?

Commenté [MB27]: What does that mean? Do you mean increase monotonically?

Commenté [MB28]: Isn't this one static for the same publisher?

Commenté [MB29]: Same receiver as well?

loss can be detected. When the last value (2^32-1) of Message ID has been ~~generated~~reached, the Message ID wraps around and restarts at 0.

Different subscribers MAY share the same Message ID sequence.

* Options ~~is~~ 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 ~~(i.e., Message ID)~~. ~~The details of the o~~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 CBOR, XML, or JSON format. ~~It is conceivable that a~~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 ~~using~~ used 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.

Implementations 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 shown~~, illustrated~~ in Figure 3.

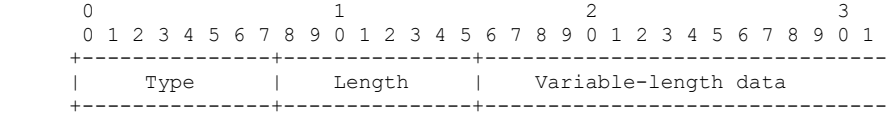

Figure 3: Generic Option Format

* Type: ~~1~~ 1-octet describing the option type~~.~~.

* Length: ~~1~~ 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~~ data~~Value~~.

Commenté [MB30]: Is there any specific action if such event happens?

Commenté [MB31]: Should this be covered by a configuration parameter?

Commenté [MB32]: A default should be called per RFC8639:

«A specification for a transport MUST identify any encodings that are supported. If a configured subscription's transport allows different encodings, the specification MUST identify the default encoding.»

a mis en forme : Surlignage

a mis en forme : Surlignage

Commenté [MB33]: Do you mean distinct encoding can be used in the same subscription session?

What is the benefit? Unless I missed the intent here, this adds complexity.

Commenté [MB34]: I guess you need an IANA registry

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

Commenté [MB35]: You may elaborate the rationale for imposing the order.

Commenté [MB36]: Do you really need this?

4.1. Segmentation Option

The UDP payload length is limited to 65527 bytes (65535 - 8 bytes).
~~Application~~ 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.

a mis en forme : Surlignage

Commenté [MB37]: May cite «IP Fragmentation Considered Fragile» (BCP 230)

~~Consequently, i~~Implementations ~~of the mechanism~~ MUST provide a configurable ~~max-segment-size option~~parameter to control the maximum size of a
UDP-Notif segment.

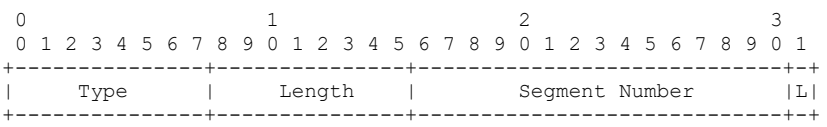

Figure 4: Segmentation Option Format

The Segmentation Option (Figure 4) is ~~to be~~ included when the message content is
~~segmented~~ fragmented into multiple segments. Different segments of one message
share the same Message ID. ~~An illustration is provided in Figure 4.~~
The fields of this ~~TLV~~ option are as follows:

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

* Length: ~~Generic option field which~~ indicates the length of this Option, in octets. It ~~is a fixed value~~MUST be set to ~~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~~ segment ~~Number~~ number value of 0. The ~~Segment~~ segment ~~Number~~ number cannot wrap around.

Commenté [MB38]: It is unlikley to exceed 32 768 segments!

I suggest to add a guard in the security section to discard segments when they exceed a certain number because this can be used as abuse to require computation resources to reassemble a large number of segments.

* L: ~~is a flag to~~ 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.

Commenté [MB39]: FYI, M(More) is used in other specs such as rfc9177.

An Iimplementations of this specification MUST NOT rely on IP fragmentation by default to carry large messages. An iImplementations of this specification MUST either restrict the size of individual messages carried over this protocol, or support the segmentation option. The implementor or user SHOULD configure the max-segment-size SHOULD be set so that the size of a UDP-Notif segment and the size of the IP layer together does do not exceed the MTU of the egress interface.

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.

The rReceivers SHOULD support the reception of unordered segments. The implementation Implementations 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. If the receiver collects a segment more than once, the implementation SHOULD drop the duplicate segment.

X

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.

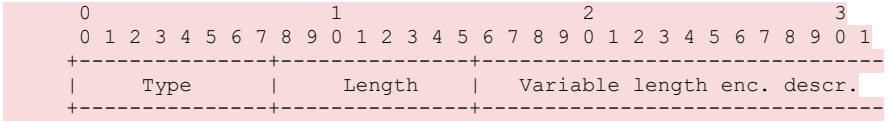

Figure 5: Private Encoding Option Format

* 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 in octets. 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 tThis section, we provides an applicability statement for the

Commenté [MB40]: Should the segment option appear first?

Commenté [MB41]: What is the value of imposing this?

Commenté [MB42]: Not sure to get the rationale here. Whether an option is present in subsequent segment may be left to the definition of an option.

a mis en forme : Surlignage

Commenté [MB43]: Concretely? What are the recommended timers/resources for waiting for missing segments? This should also be controlled to avoid DoS, etc.

Commenté [MB44]: No mention about reassembly

Commenté [MB45]: This is underspecified. I suggest to remove this from the spec.

~~proposed~~ UDP-Notif 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.

The main use case of the UDP-Notif~~proposed~~ 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 ~~proposed~~ 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 ~~QoS~~ dedicated class of services on its related flows.

As per [RFC8085], such applications ~~MAY~~ may choose not to implement any form of congestion control, but follow the following principles.

It is NOT RECOMMENDED to use the ~~proposed~~ 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~~ rate-limiting or capacity reservations.

Implementation ~~of the proposals~~ 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 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

Commenté [MB46]: Still, these may suffer from DDoS and so on.

I think that we need to have a guard to protect the network and limit at least how segments fly + add some randomness when sending subscription

Commenté [MB47]: What does that mean?

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. Implementations ~~of the current specification~~ SHOULD allow for the configuration of the MTU.

It is RECOMMENDED that the size of a Notification Message is small and segmentation does not result in segmenting the message into too much segments to avoid dropping the entire message when there is a lost segment. When a Notification Message is large, it is RECOMMENDED to use a reliable transport such as HTTPS-notif [I-D.ietf-netconf-https-notif].

5.3. Reliability

A receiver implementation ~~for this protocol~~ SHOULD ~~deal with potential loss of packets carrying a part of segmented payload, by~~ discard~~ing~~ 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 ~~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 in this document.

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].

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 number, but MAY be configurable to listen on a different port number. ~~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 number for transmitting messages.

6.1. Session L~~l~~ifecycle

This section describes the lifecycle of UDP-Notif messages when they are ~~exerypted~~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 support the denial of service countermeasures defined by DTLS 1.3 if a given deployment can ensure that DoS attacks are not a concern. When these countermeasures are used, the Receiver responds with a DTLS

Commenté [MB48]: Already mentioned. At least, you should avoid the use of the normative language

Commenté [MB49]: In reference to what?

Commenté [MB50]: The session may not be in place and this may not be known in advance.

Commenté [MB51]: Any default ?

Commenté [MB52]: I would delete

Commenté [MB53]: I would delete

Commenté [MB54]: Which one?

Commenté [MB55]: duplicate

Commenté [MB56]: If «any source» is used, but a filtering device is on path, need to make sure these will get through to a receiver

Commenté [MB57]: Should we follow «TCP stacks commonly use a randomized source port to provide this protection [RFC6056]; UDP applications should follow the same technique » from 8085?

HelloRetryRequest containing a stateless cookie. The Publisher sends 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. 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.

Commenté [MB58]: I would replace all these details with a simple ref to bcp195.

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~~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.

X DTLS/Fragmenation

Commenté [MB59]: Some considerations are need to avoid DTLS frag. Note that 9147 says:

«DTLS messages **MAY** be fragmented into multiple DTLS records. Each DTLS record **MUST** fit within a single datagram. »

Some measures can be needed:

«If the Path MTU (PMTU) cannot be discovered, Publisher MUST assume a PMTU of 1280 bytes, as IPv6 requires that every link in the Internet have an MTU of 1280 octets or greater, as specified in [RFC8200]. If IPv4 support on legacy or otherwise unusual networks is a consideration and the PMTU is unknown, Publisher MAY assume a PMTU of 576 bytes for IPv4 datagrams (see Section 3.3.3 of [RFC1122]). » (grabbed from RFC9132)

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

7.1. YANG to configure 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 ~~model proposed in this document~~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~~YANG proposed in this document~~.

```
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-supported}?
          +--rw local-port?            inet:port-number
          |       {local-binding-supported}?
          +--rw dtls! {dtls13}?
          |  +--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 (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 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 keepalives {tls-client-keepalives}?
|     +--rw peer-allowed-to-send?   empty
|     +--rw test-peer-aliveness!
|        +--rw max-wait?       uint16
|        +--rw max-attempts?   uint8
+--rw enable-segmentation?   boolean {segmentation}?
+--rw max-segment-size?      uint16 {segmentation}?
```

Commenté [MB60]: The spec does describes how this is used for UDP-Notif

Commenté [MB61]: I would a rate to limit the number of segment/s (and other parameters to reflect other comments above)

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_~~-grouping~~" defines the necessary parameters to configure the transport defined in this document using the generic "udp-client_~~-grouping~~" grouping from the "ietf-udp-client" module [I-D.ahuang-netconf-udp-client-server] and the "tls-client_~~-grouping~~" defined in the "ietf-tls-client" module [I-D.ietf-netconf-tls-client-server].

Commenté [MB62]: No need to suffix a grouping!

Commenté [MB63]: Hmm!

This should be refreshed

Commenté [MB64]: RFC 9645 - YANG Groupings for TLS Clients and TLS Servers

```
<CODE BEGINS> file "ietf-udp-notif-transport@2024-10-17.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-udp-client {
    prefix udpc;
    reference
      "RFC ZZZZ: YANG Grouping for UDP Clients and UDP Servers";
  }
  import ietf-tls-client {
```

```
      prefix tlsc;
      reference
        "RFC ~~TTTT~~9645: YANG Groupings for TLS Clients and TLS Servers";
    }

    import ietf-tls-common {
      prefix tlscmn;
      reference
        "RFC ~~TTTT~~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:  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 a model for configuring UDP-Notif as a transport
      for c~~C~~onfigured YANG ~~Subscriptions~~subscriptions.

      Copyright (c) ~~2024~~ 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; see the RFC
      itself for full legal notices.";

    revision 2024-10-17 {
      description
        "Initial revision";
      reference
        "RFC XXXX: UDP-based Transport for Configured YANG
Subscriptions";
    }

   /*

    * FEATURES
    */
```

```
      feature encode-cbor {
        description
          "This feature iIndicates that CBOR encoding of notification
           messages is supported.";
        reference
          "RFC 9254: CBOR Encoding of Data Modeled with YANG";

      }
      feature dtls13 {
        description
          "This feature iIndicates that DTLS 1.3 encryption of UDP
           packets is supported.";
      }
      feature segmentation {
        description
          "This feature iIndicates segmentation of notification messages
          is supported.";
        reference
          "RFC XXXX: UDP-based Transport for Configured YANG
Subscriptions";

      }

     /*
      * 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 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 identification of a UDP-Notif
target
           receiver.";

        uses udpc:udp-client-grouping_ {
          refine remote-port {
            mandatory true;
          }
        }

        container dtls {
```

Commenté [MB65]: Add a reference statement (RFC9254)

Commenté [MB66]: Add a reference statement to 9147

```
    if-feature dtls13;
    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";
      }
      refine "hello-params/tls-versions/min" {
        must "not(derived-from-or-self(current(), "
          + "'tlscmn:tls12'))" {
          error-message
                "TLS 1.2 is not supported as min TLS version";
        }
      }
      refine "hello-params/tls-versions/max" {
        must "not(derived-from-or-self(current(), "
          + "'tlscmn:tls12'))" {
          error-message
                "TLS 1.2 is not supported as max TLS version";
        }
      }
    }
    description
      "Container for configuring DTLS 1.3 parameters.";
  }

  leaf enable-segmentation {
    if-feature segmentation;
    type boolean;
    default true;
    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 uint16;
    description
      "UDP-Notif provides a configurable max-segment-size to
      control the size of each segment (UDP-Notif header, with
      options, included).";
  }
}

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

8. IANA Considerations

~~This document describes several new registries, the URIs from IETF XML Registry and the registration of a new YANG module name.~~

8.1. IANA Rregistries

This document ~~is~~ requests IANA to ~~creating~~create a new registry group called "UDP-Notif protocol".

Also, this document requests IANA to create the following ~~3~~ three registries called "UDP-Notif media types", "UDP-Notif option types", and "UDP-Notif header version" under the new group "UDP-Notif protocol", under the "UDP-Notif protocol" registry group.

The registration procedure ~~is made using~~for all these registries follows the "Standards Action" ~~process~~ policy (Section 4.9 of~~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: RFC-to-be

Value: 1
Description: media type application/yang-data+json
Reference: ~~<xref target="~~RFC_8040~~"/>~~

Value: 2
Description: media type application/yang-data+xml
Reference: ~~<xref target="~~RFC_8040~~"/>~~

Value: 3
Description: media type application/yang-data+cbor
Reference: ~~<xref target="~~RFC_9254~~"/>~~

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: RFC-to-be

Value: TBD1 (suggested value: 1)
Description: Segmentation Option
Reference: RFC-to-be

Value: TBD2 (suggested value: 2)
Description: Private Encoding Option
Reference: RFC-to-be

a mis en forme : Surlignage

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: UDP based Publication Channel for Streaming Telemetry
Reference: draft-ietf-netconf-udp-pub-channel-05

Value: 1
Description: UDP-based Transport for Configured YANG Subscriptions.
Reference: RFC-to-be

8.2. URI

IANA is also requested to assign a ~~two~~ 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.

8.3. YANG Mmodule ~~name~~Name

This document also requests a new YANG module ~~names~~ name in the YANG Module Names registry [~~RFC8342~~RFC6020] with the following suggestions:

name: ietf-udp-notif-transport
namespace: urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport
maintained by IANA? N
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 has been integrated into the Pmacct open source Network Telemetry data collection.

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, an encryption layer MUST be implemented for non secured networks. A specification of UDP-notif using DTLS 1.3 as its encryption layer is presented in Section 6.

XX

Commenté [MB67]: Add security cons for the YANG module.

11. 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, Yunan Gu, Joseph Touch, Andy Bierman and Carsten Bormann for their constructive suggestions for improving

this document.

12. References

12.1. Normative References

[I-D.ahuang-netconf-udp-client-server]
Feng, A. H., Francois, P., and K. Watsen, "YANG Grouping for UDP Clients and UDP Servers", Work in Progress, Internet-Draft, draft-ahuang-netconf-udp-client-server-01, 21 January 2024, <https://datatracker.ietf.org/doc/html/draft-ahuang-netconf-udp-client-server-01>.

[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-10, 18 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-distributed-notif-10>.

[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-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-41, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-41>.

[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>.

~~[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.~~

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013,

a mis en forme : Surlignage

Commenté [MB68]: Not used in the doc

<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>.

[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>.

[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>.

[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>.

12.2. Informative References

[I-D.ahuang-netconf-notif-yang] Feng, A. H., Francois, P., Graf, T., and B. Claise, "YANG model for NETCONF Event Notifications", Work in Progress, Internet-Draft, draft-ahuang-netconf-notif-yang-05, 18 June 2024, <https://datatracker.ietf.org/doc/html/draft-

a mis en forme : Surlignage

ahuang-netconf-notif-yang-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>.

[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>.

[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>.

[RFC7923] Voit, E., Clemm, A., and A. Gonzalez Prieto, "Requirements for Subscription to YANG Datastores", RFC 7923, DOI 10.17487/RFC7923, June 2016, <https://www.rfc-editor.org/info/rfc7923>.

[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>.

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">
          <remote-address>192.0.5.1</remote-address>
```

```
          <remote-port>12345</remote-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">
          <remote-address>192.0.5.1</remote-address>
          <remote-port>12345</remote-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": "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: JSON Payload to be sent

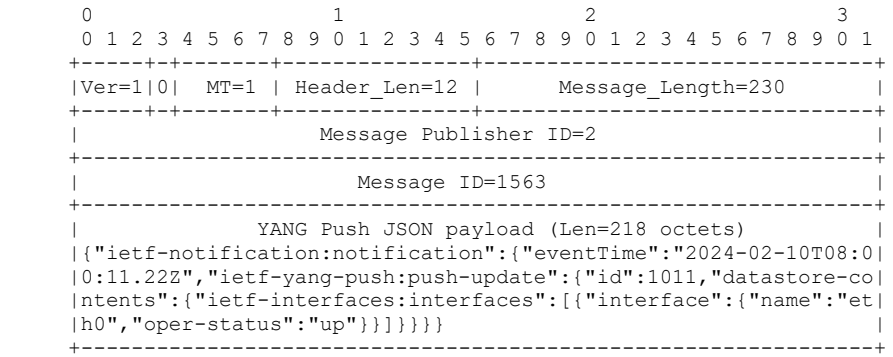

Figure 7: UDP-Notif transport message

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