Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 10 August 2026

A. Huang-Feng
P. Francois
INSA-Lyon
T. Graf
Swisscom
B. Claise
Everything OPS

6 February 2026

Extensible YANG Model for YANG-Push Notifications
draft-ietf-netconf-notif-envelope-04

Abstract

This document defines a new extensible Notification structure, defined in YANG, for use in YANG-Push Notification messages, both for NETCONF and RESTCONF, enabling any YANG-compatible encodings such as XML, JSON, or CBOR. Additionally, it defines two essential extensions to this structure, the support of a hostname and a sequence number and the support of a timestamp characterizing the moment when the data was observed.

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 10 August 2026.

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Terminology 4
2. Relationship to past documents 4
2.1. Relationship to RFC5277 4
2.2. Relationship to RFC8639 5
2.3. Relationship to RFC7950 5
2.4. Relationship to RFC7951 5
2.5. Relationship to RFC9254 6
3. Notification Envelope Model 6
3.1. Notification Envelope Structure 7
3.1.1. Base Notification Model 7
3.1.2. Encodings of the Notification Envelope 7
3.2. Extensions for the Notification Envelope 14
3.2.1. Support of Hostname and Sequencing 14
3.3. Extensions for the YANG-Push Header 16
3.3.1. Support of Observation Timestamp 16
4. Enabling the Notification Envelope 22
5. Discovering the Support of the Notification Envelope 23
6. Operational Considerations 24
7. YANG Modules 24
7.1. The 'ietf-yp-notification' Module 25
7.1.1. YANG 'ietf-yp-notification' Tree Diagram 25
7.1.2. YANG 'ietf-yp-notification' Module 25
7.2. The 'ietf-yp-observation' Module 30
7.2.1. YANG 'ietf-yp-observation' Tree Diagram 30
7.2.2. YANG 'ietf-yp-observation' Module 30
8. Implementation Status 33
8.1. Huawei VRP 34
8.2. 6WIND VSR 34
8.3. Cisco IOS XR 34
9. Security Considerations 34
10. IANA Considerations 35
10.1. URI 35
10.2. YANG module name 35
10.3. YANG SID-file 36
11. Acknowledgements 36
12. References 36
12.1. Normative References 36
12.2. Informative References 38
Appendix A. .sid file 39

Appendix B. Example extending the Notification Envelope 43
Authors' Addresses 45

1. Introduction

YANG-Push [RFC8639] allows Publishers to send Notifications to a data collection system. The YANG-Push Receiver decodes the message and optionally validates the header and the content before forwarding to the next process in the data collection system.

The Notification encoded message from YANG-Push, as defined in Section 2.6 of [RFC8639], is currently based on the XML model from NETCONF Event Notifications [RFC5277]. This model has the drawback that only a single mandatory 'eventTime' leaf is defined and does not offer a way to extend this header with new Notification metadata. Additionally, the 'eventTime' reports "The time the event was generated by the event source" and not the metric observation time; the data collection system might draw the wrong conclusion in case of a busy Publisher that delays the YANG-Push export or in case a YANG-Push Publisher re-exports the YANG-Push message. Finally, this XML model is only valid for XML-based environments. When messages are encoded in other YANG encodings, such as JSON [RFC7951] or CBOR [RFC9254], validators cannot use YANG to validate the message schema.

YANG data consumers receiving Notifications require additional Notification metadata to understand the full context of the received message. For example, in addition to the timestamp of when the event was encoded, it is also important to know the timestamp when the metrics were observed, the hostname that sourced the message, and have sequence numbers. The hostname is required because transport-level source information is not preserved once Notifications are forwarded to downstream systems, and having sequence numbers at the notification level enable operators to detect lost notifications throughout the data processing chain. This additional Notification metadata is also helpful in correlating the data with other sources of Network Telemetry [RFC9232] information.

For such reasons, this document proposes the following:

* First, it provides an extensible YANG Notification header allowing implementors and authors of Internet-Drafts to easily add new Notification metadata to the Notification message.

* Second, it provides the first crucial extensions enabling operators to identify which network node publishes which YANG-Push messages, when the events or metrics were observed on the network node, and in which order the messages were transmitted to the Receiver.

* And finally, it provides a way to enable and disable these extensions globally at the Server, making the coexistence of different YANG-Push and NETCONF Event Notification [RFC5277] possible.

1.1. 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 terms "Subscription", "Dynamic Subscription", "Configured Subscription", "Publisher", "Receiver" and "Notification Message" are used as defined in [RFC8639].

The term "Client" and "Server" is used as defined in [RFC6241] for NETCONF and [RFC8040] for RESTCONF.

The terms "Implementation-time Information" and "Runtime Information" are used as defined in [RFC9196].

In addition, this document defines the following terms:

Notification Metadata: Additional data describing the context of a notification that is sent in each message, e.g. which node generated the message or at which time the notification was published.

Notification Envelope: YANG structure encapsulating the payload of a notification, allowing the inclusion of metadata.

2. Relationship to past documents

This section shows the relationship to [RFC5277], [RFC8639], [RFC7951] and [RFC9254].

2.1. Relationship to RFC5277

[RFC5277] defines a mechanism for NETCONF Servers to send Notifications to collectors. These are the key relationships between the current document and [RFC5277]:

* This document does not change the header defined by [RFC5277] nor update any behavior defined in [RFC5277]. Implementations of [RFC5277] use the header defined in Section 2.2.1 of [RFC5277].

* The co-existence of the Notification model defined in [RFC5277] and the model defined in the current document is possible. The co-existence is discussed in Section 6.

2.2. Relationship to RFC8639

Subscribed Notifications [RFC8639] defines a mechanism on top of [RFC5277] to stream Notifications from the Server. These are the key relationships between the current document and [RFC8639]:

* Section 1.4 of [RFC8639] states that the solution uses the Notification header defined in [RFC5277]. This document proposes a new header, which Clients can enable to replace the header defined in [RFC5277] for YANG-defined event records. When this new header is used, YANG-defined Notifications sent via Subscribed Notifications [RFC8639] are encoded as defined in Section 3.1. Servers MUST continue using the [RFC5277] header for NETCONF event streams.

* 2.4 and 2.5 of [RFC8639] defines how a Dynamic Subscription and Configured Subscriptions are managed. This document extends the Subscribed Notifications configuration [RFC8639] to support enabling the new header defined in this document.

2.3. Relationship to RFC7950

[RFC7950] defines how YANG data is encoded in XML. These are the key relationship points between the current document and [RFC7950]:

* Section 7.16.2 of [RFC7950] defines the XML encoding of YANG Notification. This document defines a new header for such Notifications. When a YANG-Push Publisher implements the specifications in this document with the XML encoding, the Notifications are encoded according to Section 3.1.2.1.

2.4. Relationship to RFC7951

[RFC7951] defines how YANG data is encoded using JSON. These are the key relationship points between the current document and [RFC7951]:

* [RFC7951] does not define explicitly how a YANG Notification should be encoded using JSON encoding. This document specifies a new header for such Notifications. When a YANG-Push Publisher implements the specifications in this document with JSON encoding, the Notifications are encoded according to Section 3.1.2.2.

2.5. Relationship to RFC9254

[RFC9254] defines how YANG data is encoded using CBOR. These are the key relationship points between the current document and [RFC9254]:

* [RFC9254] does not define explicitly how a YANG Notification should be encoded using CBOR encoding. When a YANG-Push Publisher implements the specifications in this document in CBOR encoding, the Notifications are encoded according to Section 3.1.2.3 in this document.

3. Notification Envelope Model

Section 4.2.10 of [RFC7950] defines the encoding of YANG Notifications. A Notification is defined by a 'notification' statement in the YANG module. When a NETCONF Server sends a Notification, it comprises two parts: a header containing Notification metadata that encapsulates the content and the content defined by the 'notification' statement.

In YANG 1.1 [RFC7950], the Notification header is based on the model defined in [RFC5277] which contains a single metadata 'eventTime' leaf. An example extracted from [RFC7950] is shown in the following XML:

```
<notification
    xmlns="urn:ietf:params:netconf:capability:notification:1.0">
  <eventTime>2007-09-01T10:00:00Z</eventTime>
  <link-failure xmlns="urn:example:system">
    <if-name>so-1/2/3.0</if-name>
    <if-admin-status>up</if-admin-status>
    <if-oper-status>down</if-oper-status>
  </link-failure>
</notification>
```

This document defines a new Notification header and enables extending this header with new Notification metadata. The Notification header and extensions in this document are to be used in YANG-Push [RFC8641] environments and can be implemented with NETCONF [RFC6241] and RESTCONF [RFC8040]. When enabled, this new header MUST replace all Notifications defined in both Subscribed Notifications [RFC8639] and YANG-Push [RFC8641] for the entire Server, for both Configured and Dynamic Subscriptions.

The remainder of this section is structured as follows. Section 3.1 defines the Notification Envelope and how this structure is encoded using XML, JSON, and CBOR. Section 3.2 and Section 3.3 defines the first Notification extensions to both the Notification Envelope and the YANG-Push header.

The Notification Envelope and extensions are enabled through the mechanism defined in Section 4. Clients can discover the support of this header and extensions for both implementation-time and runtime through the discovery mechanism defined in Section 5.

3.1. Notification Envelope Structure

This section defines the structure of the Notification Envelope. The following sections define how this structure is encoded in XML, JSON and CBOR.

3.1.1. Base Notification Model

When a YANG-Push Publisher uses the Notification model defined in this document, the Notification is structured as follows:

* The Notification is encapsulated in a root ’envelope’ container.

* The header of the Notification contains the Notification metadata that is enabled during the configuration of the Subscription as child nodes of the root ’envelope’ container.

* The content of the Notification defined by the ’notification’ statement is encoded in the ’contents’ leaf.

The following YANG tree [RFC8340] illustrates the Notification Envelope supporting only the mandatory metadata ’event-time’. Refer to Section 3.2 for more extensions to this header.

```
  structure envelope:
    +-- event-time         yang:date-and-time
    +-- contents?          <anydata>
```

3.1.2. Encodings of the Notification Envelope

The YANG Notification can be encoded using XML [W3C.REC-xml-20001006][RFC7951], JSON [RFC7951] and CBOR [RFC9254].

3.1.2.1. XML encoding

A YANG Notification encoded in XML is structured as a root 'envelope' container. The namespace of this container is the namespace defined in the YANG module 'ietf-yp-notification':

urn:ietf:params:xml:ns:yang:ietf-yp-notification

Two mandatory child nodes within the 'envelope' container are expected, representing the event time and the Notification payload.

When other Notification metadata is enabled through configuration, the supplementary nodes are encoded at the same level as the mandatory 'event-time' node. The YANG nodes in the Notification header use the XML namespace from the module that defines the nodes. This document defines two Notification metadata. Refer to Section 3.2 for more details.

The content of the Notification that is defined by the 'notification' statement is encoded in the 'contents' node. The name and namespace of this payload element are determined by the YANG module containing the 'notification' statement representing the Notification message.

The following example shows a 'push-update' Notification defined in the YANG module of YANG-Push [RFC8641] encoded in XML:

```
<envelope xmlns="urn:ietf:params:xml:ns:yang:ietf-yp-notification">
  <event-time>2024-10-10T10:59:55.32Z</event-time>
  <contents>
    <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
      <id>1011</id>
      <datastore-contents>
        <interfaces
          xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
          <interface>
            <name>eth0</name>
            <type>iana-if-type:ethernetCsmacd</type>
            <if-index>1</if-index>
            <admin-status>down</admin-status>
            <oper-status>down</oper-status>
            <statistics>
              <discontinuity-time>
                2013-04-01T03:00:00+00:00
              </discontinuity-time>
            </statistics>
         </interface>
        </interfaces>
      </datastore-contents>
    </push-update>
  </contents>
</envelope>
```

Figure 1: XML-encoded notification

3.1.2.2. JSON Encoding

A YANG Notification encoded in JSON is structured as a root 'envelope' container. The namespace of this container is the name of the YANG module 'ietf-yp-notification' defined in Section 7.1.2.

Two mandatory child nodes within the 'ietf-yp-notification:envelope' container are expected, representing the event time and the Notification payload. The 'event-time' node is defined within the same namespace as the 'ietf-yp-notification:envelope' container.

When other Notification metadata is enabled through configuration, the supplementary nodes are encoded at the same level as the mandatory 'event-time' node. The YANG nodes in the Notification header use the YANG module name from the module that defines the nodes as its namespace. Two additional metadata are described in this document. Refer to Section 3.2 for more details.

The content of the Notification that is defined by the 'notification' statement is encoded in the 'contents' node. The name and namespace of this payload element are determined by the YANG module containing the 'notification' statement representing the Notification message.

The following example shows a "push-update" Notification defined in the YANG module of YANG-Push [RFC8641] encoded in JSON:

```
{
  "ietf-yp-notification:envelope": {
    "event-time": "2024-10-10T08:00:11.22Z",
    "contents": {
      "ietf-yang-push:push-update": {
        "id": 1011,
        "datastore-contents": {
          "ietf-interfaces:interfaces": {
            "interface": [
              {
                "name": "eth0",
                "type": "iana-if-type:ethernetCsmacd",
                "if-index": 1,
                "admin-status": "up",
                "oper-status": "up",
                "statistics": {
                  "discontinuity-time": "2024-10-10T07:50:00Z"
                }
              }
            ]
          }
        }
      }
    }
  }
}
```

Figure 2: JSON-encoded Notification

3.1.2.3. CBOR Encoding

YANG Notifications can be encoded in CBOR using Names or SIDs in keys.

Notifications encoded using names is similar to JSON encoding as defined in Section 3.3 of [RFC9254]. The key of the element can be the name of the element itself or be namespace-qualified. In the latter case, the namespace of the Notification encoded message uses the YANG module name 'ietf-yp-notification', defined in Section 7.1.2.

Notification encoded using YANG-SIDs replaces the name of the keys of the CBOR encoded message with a 63-bit unsigned integer. In this case, the keys of the encoded data use the SID value as defined in Section 3.2 of [RFC9254]. A SID allocation process is needed beforehand as defined in [RFC9595].

In the Notification, two mandatory child nodes within the 'ietf-yp-notification:envelope' container are expected, representing the event time and the Notification payload. The 'event-time' node is defined within the same namespace as the 'ietf-yp-notification:envelope' container.

When other Notification metadata is enabled through configuration, the supplementary nodes are encoded at the same level as the mandatory 'event-time' node. The YANG nodes in the Notification header use the YANG module name from the module that defines the nodes as its namespace when they are encoded as names. When encoded using YANG SIDs, the SID value assigned to the metadata node is used. A .sid file requesting the SID values for the metadata defined in this document can be found in Appendix A. The .sid file includes the metadata defined in Section 3.2.

The content of the Notification that is defined by the 'notification' statement is encoded in the 'contents' node. The name and namespace of this payload element are determined by the YANG module containing the 'notification' statement representing the Notification message. Similarly, SIDs can be used as keys if they are allocated following the process defined in [RFC9595].

Figure 3 shows a 'push-update' Notification defined in the YANG module of YANG-Push [RFC8641] encoded in CBOR using names as keys. The example uses the CBOR diagnostic notation as defined in Section 3.1 of [RFC9254]:

```
{
  "ietf-yp-notification:envelope": {
    "event-time": "2024-10-10T08:00:11.22Z",
    "contents": {
      "ietf-yang-push:push-update": {
        "id": 1011,
        "datastore-contents": {
          "ietf-interfaces:interfaces": {
            "interface": [
              {
                "name": "eth0",
                "type": "iana-if-type:ethernetCsmacd",
                "if-index": 1,
                "admin-status": "up",
                "oper-status": "up",
                "statistics": {
                  "discontinuity-time": "2024-10-10T07:50:00Z"
                }
              }
            ]
          }
        }
      }
    }
  }
}
```

Figure 3: CBOR-encoded Notification using diagnostic notation

Figure 4 shows the same Notification encoded using SIDs:

```
{
  2957: {                                / ietf-yp-notification:envelope (SID 2957)
/
    2: "2024-10-10T08:00:11.22Z",        / event-time (SID 2959) /
    1: {                                 / contents (SID 2958) /
      "ietf-yang-push:push-update": {
        "id": 1011,
        "datastore-contents": {
          "ietf-interfaces:interfaces": {
            "interface": [
              {
                "name": "eth0",
                "type": "iana-if-type:ethernetCsmacd",
                "if-index": 1,
                "admin-status": "up",
                "oper-status": "up",
                "statistics": {
                  "discontinuity-time": "2024-10-10T07:50:00.00Z"
                }
              }
            ]
          }
        }
      }
    }
  }
}
```

Figure 4: CBOR-encoded Notification using YANG SIDs in CBOR diagnostic notation

Note to the RFC-Editor: Please change the SID values from the example Figure 4 with the ones allocated by IANA.

Note that in the example shown in Figure 4, the Notification payload uses names as keys. These keys can also be encoded as SIDs. The corresponding SID values must be allocated in the IANA registry, following the procedures defined in [RFC9595].

When SIDs are used throughout the Notification Envelope, they are encoded as deltas relative to the parent node by default. The absolute SID can also be used using the tag 47. Refer to [RFC9254] for more details.

3.2. Extensions for the Notification Envelope

This section describes two optional YANG nodes for the Notification Envelope header: a 'hostname' and a 'sequence-number', which are enabled by default when using the Notification Envelope defined in this document. The Client discovers the support of these two optional leafs with the mechanism defined in Section 5.

The Notification Envelope, including the metadata, results in the following YANG tree [RFC8340].

```
  structure envelope:
    +-- event-time         yang:date-and-time
    +-- hostname?          inet:host-name {hostname-sequence-number}?
    +-- sequence-number?   yang:counter32 {hostname-sequence-number}?
    +-- contents?          <anydata>
```

3.2.1. Support of Hostname and Sequencing

When YANG-Push Notification messages are forwarded from a Receiver to another system, such as a message broker or a time series database, the transport context is lost since it is not part of the Notification metadata of the Notification encoded message. Therefore, the downstream system is unable to associate the message with the publishing process (the exporting network node), nor able to detect message loss or reordering.

To correlate network data among different Network Telemetry planes as described in Section 3.1 of [RFC9232] or among different YANG-Push Subscription types as defined in Section 3.1 of [RFC8641], a reference to the node streaming the data is needed. This is essential for understanding the timely relationship among these different planes and YANG-Push Subscription types.

Today, network operators work around this impediment by preserving the transport source IP address and sequence numbers of the publishing process. However, this implies encoding this information in the YANG-Push Notification messages which impact the semantic readability of the message in the downstream system.

On top of that, the transport source IP address might not represent the management IP address by which the YANG-Push Publisher should be known. In other terms, the 'source-host' [RFC6470], which is the "Address of the remote host for the session" might not be the management IP address.

To overcome these issues, this document defines an extension to the Notification Envelope to encode a hostname and a sequence number. This allows the downstream system to not only be able to identify from which network node, Subscription, and time the message was published but also, the order of the published messages.

hostname: Describes the hostname of the network node that published the message. The hostname uniquely identifies the node within the network.

sequence-number: Generates a unique sequence number for each published message by the Publisher process. The initial number is 1 and counts up by 1 at every published Notification message until it reaches 4294967295. Then, it wraps around and restarts at 0. The value 0 is used to detect wrap arounds.

Figure 5 provides an example of a JSON encoded, [RFC8259], 'push-update' Notification message with 'hostname' and 'sequence-number' included.

```
{
    "ietf-yp-notification:envelope": {
    "event-time": "2023-03-25T08:30:11.22Z",
    "hostname": "example-router.example.com",
    "sequence-number": 1,
    "contents": {
      "ietf-yang-push:push-update": {
        "id": 6666,
        "datastore-contents": {
          "ietf-interfaces:interfaces": {
            "interface": [
              {
                "name": "eth0",
                "type": "iana-if-type:ethernetCsmacd",
                "if-index": 1,
                "admin-status": "up",
                "oper-status": "up",
                "statistics": {
                  "discontinuity-time": "2023-03-25T08:20:00.00Z"
                }
              }
            ]
          }
        }
      }
    }
  }
```

Figure 5: JSON Example for a 'push-update' Notification message

3.3. Extensions for the YANG-Push Header

This section described two optional 'push-update' and 'push-change-update' Notification header extensions which are enabled by default when using the Notification Envelope defined in this document. The Client discovers the support of these two leafs with the mechanism defined in Section 5.

This document defines the following Notification metadata as shown in the following YANG tree [RFC8340]. See the following sections for more details.

```
module: ietf-yp-observation

  augment /yp:push-update:
    +--ro timestamp?       yang:date-and-time
    +--ro point-in-time?   enumeration
  augment /yp:push-change-update:
    +--ro timestamp?       yang:date-and-time
    +--ro point-in-time?   enumeration
```

3.3.1. Support of Observation Timestamp

To correlate network data among different Network Telemetry planes, as described in Section 3.1 of [RFC9232], or among different YANG-Push Subscription types, as defined in Section 3.1 of [RFC8641], a Receiver needs a timestamp reference to align all the metrics and events. The observation timestamp defined in this document characterizes the moment the state change was observed or the moment when the data was measured, so that a Receiver can correctly align the collected data.

The delay between the YANG-Push export process and the reception of the message at the Receiver instance can be measured using the node 'event-time' defined in Section 3.1.1. However, as the 'event-time' node only establishes the moment when the YANG-Push message was crafted and sent, the moment when such metrics were collected or the state changes were observed cannot be measured using this timestamp. The observation timestamp defined in this section characterizes the moment when the metrics were observed, which enable aligning the received metrics to the actual moment they were measured.

When the time bucket length in a time series database and the periodic YANG-Push Subscription time are configured with the same values, the 'event-time' of the NETCONF Notification message header can be used for indexing the data in the time series database. There

is a variable delay between the observation timestamp, the 'event-time', and the 'anchor-time' as described in Section 4.2 of [RFC8641]. When these timestamps are close to the time bucket boundaries, a time bucket may experience data collection discrepancies, e.g. 0 measurements are aggregated into one time bucket while the next time bucket contains 2 measurements. This leads to inconsistent accounting errors in the time series database. This problem is resolved using the observation timestamp instead of the 'event-time' for time series database indexation.

By extending YANG-Push Notifications with the observation timestamp and a 'point-in-time' node, the data collection process can always ensure it has the best available time for indexing the data. It can therefore use unconditionally the observation timestamp node in the data processing chain to correctly align the metrics and events. At the same time, the 'point-in-time' node ensures that the semantics associated with the timestamp are not lost throughout the data processing chain.

Besides the Subscription ID as described in Section 3.7 of [RFC8641], the following network observation time metadata objects are part of 'push-update' and 'push-change-update' Notifications.

timestamp: States the measurement observation time for the 'push-update' Notification in 'periodic' Subscriptions and for the 'push-change-update' Notification in 'on-change' Subscriptions.

By comparing the observation timestamp of two 'push-update' Notifications in a periodic Subscription, the collector can deduce the actual cadence of the measurements, and compare it with the Subscription configuration. In case of an 'on-change' Subscription it states the time when the network state change was observed.

point-in-time: The enumeration states at which point in time the value of the observation timestamp was observed. Choices are:

'current-accounting' states the point in time where the metrics are polled and observed in "periodic" Subscriptions.

'initial-state' states the initial point in time when the Subscription was established and the state was observed for 'on-change sync on start' Subscriptions.

'state-changed' states the point in time when the state change was observed after the Subscription was established for 'on-change' and 'on-change sync on start' Subscriptions.

3.3.1.1. Usage Example

This section illustrates the usage of the 'point-in-time' node in two different Subscriptions. Section 3.3.1.1.1 showcases a YANG-Push Subscription monitoring the state of an interface using an 'on-change sync on start' Subscription. Section 3.3.1.1.2 illustrates the usage of the 'point-in-time' node within periodic Subscriptions.

3.3.1.1.1. On-Change Subscriptions

Figure 6 illustrates the set of events that lead to the generation of 'on-change' YANG-Push Notifications. This timeline depicts the following states and events:

* T1: At first, the operational state of the interface is "Up". The Subscription is not configured yet at this stage and thus, Notifications are not triggered for this state change.

* T2: After configuring an 'on-change' Subscription supporting 'sync on start', the Publisher sends the initial state of the interface. The initial state is polled based on the event having happened at T1.

* T3: This is the moment the interface changes its operational status to "Down".

* T4: After the interface state changes at T3, the Publisher generates the 'on-change' Notification alerting the Receiver.

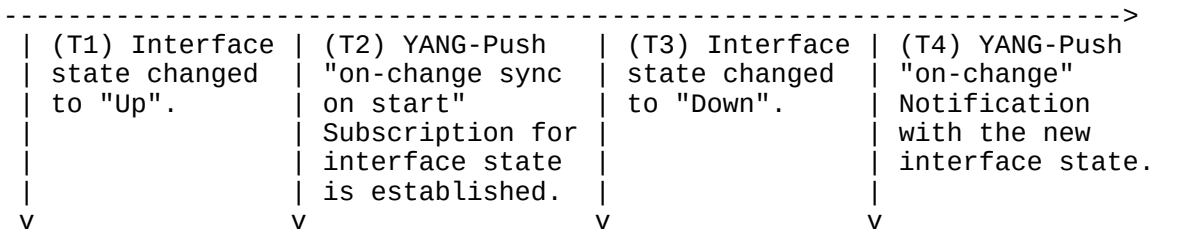

Figure 6: Example timeline for On-Change Sync on Start Subscription

At T2, after configuring the Subscription, the Publisher triggers a 'push-update' Notification as depicted in Figure 7. The 'event-time' is the moment the YANG-Push process generates the Notification (T2). The 'ietf-yp-observation:timestamp' node characterizes the moment of the interface changed it status to "Up" (T1). T1 is the latest moment this state was observed by the process. In this case, the 'point-in-time' node is set to 'initial-state'.

```
{
  "ietf-yp-notification:envelope": {
    "event-time": "2025-03-25T08:30:11.22Z",
    "hostname": "example-router.example.com",
    "sequence-number": 1,
    "contents": {
      "ietf-yang-push:push-update": {
        "id": 6666,
        "ietf-yp-observation:timestamp": "2025-03-25T08:29:30.22Z",
        "ietf-yp-observation:point-in-time": "initial-state",
        "datastore-contents": {
          "ietf-interfaces:interfaces": {
            "interface": [{
              "name": "eth0",
              "type": "iana-if-type:ethernetCsmacd",
              "if-index": 1,
              "admin-status": "up",
              "oper-status": "up",
              "statistics": {
                "discontinuity-time": "2025-03-25T06:43:12Z"
              }
            }]
          }
        }
      }
    }
  }
}
```

Figure 7: Example of 'push-update' Notification sent after the Subscription is established.

After T3, the Publisher triggers a 'push-change-update' Notification announcing an interface status change to the Receiver as depicted in Figure 8. In this case, the 'ietf-yp-observation:timestamp' node characterizes the moment the interface changed its status at T3. The value of the 'event-time' node characterizes the moment the YANG-Push process generated the Notification at T4.

```
{
  "ietf-yp-notification:envelope": {
    "event-time": "2025-03-25T08:35:12.22Z",
    "hostname": "example-router.example.com",
    "sequence-number": 1,
    "contents": {
      "ietf-yang-push:push-change-update": {
        "id": 2222,
        "ietf-yp-observation:timestamp": "2025-03-25T08:34:05.22Z",
        "ietf-yp-observation:point-in-time": "state-changed",
        "datastore-contents": {
          "yang-patch": {
            "patch-id": "52",
            "edit": {
              "edit-id": "edit_example_1",
              "operation": "replace",
              "target": "/ietf-interfaces:interfaces",
              "value": {
                "ietf-interfaces:interfaces": {
                  "interface": [{
                    "name": "eth0",
                    "type": "iana-if-type:ethernetCsmacd",
                    "if-index": 1,
                    "admin-status": "up",
                    "oper-status": "down",
                    "statistics": {
                      "discontinuity-time": "2025-03-25T06:43:12Z"
                    }
                  }]
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Figure 8: JSON Push Example for a push-change-update Notification message

3.3.1.1.2. Periodic Subscriptions

In periodic Subscription, the observation time characterizes the time when the metrics were polled from the datastore before it generates the actual YANG-Push message.

Figure 9 illustrates the delays associated with the generation of the YANG-Push message. The timeline shows the following states:

* T1: This is the moment the periodic Subscription is configured to push the operational status of the interface every N interval.

* T2: This is the moment the YANG-Push process polls the state data and metrics from the datastore. At this stage, the Notification is not built nor sent yet.

* T3: This is the moment the YANG-Push process generates the 'push-update' Notification and sends it to the Receiver.

At every N interval, the process repeats the steps from T2 and T3, first polling the datastore to retrieve the data (T2) and then building and sending the Notification to the Receiver (T3).

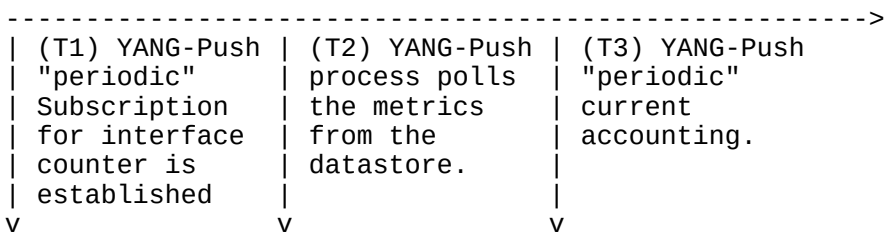

Figure 9: Example timeline for Periodic Subscription

An example of a 'push-update' Notification is illustrated in Figure 10. This example represents a message sent at T3, after the process built the YANG-Push Notification. The node 'event-time' characterizes the moment the YANG-Push process generated the message (T3). The node 'ietf-yp-observation:timestamp' establishes the moment when the metrics were polled from the datastore (T2). For these periodic Subscriptions, the 'point-in-time' node must be set to 'current-accounting'.

```
{
  "ietf-yp-notification:envelope": {
    "event-time": "2023-03-25T08:30:12.25Z",
    "hostname": "example-router.example.com",
    "sequence-number": 1,
    "contents": {
      "ietf-yang-push:push-update": {
        "id": 6666,
        "ietf-yp-observation:timestamp": "2023-03-25T08:30:00.00Z",
        "ietf-yp-observation:point-in-time": "current-accounting",
        "datastore-contents": {
          "ietf-interfaces:interfaces": {
            "interface": [{
              "name": "eth0",
              "type": "iana-if-type:ethernetCsmacd",
              "if-index": 1,
              "admin-status": "up",
              "oper-status": "up",
              "statistics": {
                "discontinuity-time": "2023-03-25T07:43:12Z",
                "in-octets": 200,
                "in-octets": 250
              }
            }]
          }
        }
      }
    }
  }
}
```

Figure 10: JSON Push Example for a push-update Notification message

4. Enabling the Notification Envelope

The Notification Envelope defined in this document can be enabled either prior to or within the same NETCONF transaction that configures the YANG-Push Subscriptions. This document augments the 'ietf-subscribed-notification' model [RFC8639] to support the configuration of the Notification Envelope.

When the node 'enable-notification-envelope' is enabled, all the Notifications defined in Subscribed Notification [RFC8639] and YANG-Push [RFC8641] are encoded as defined in Section 3.1. If additionally, any Notification metadata is enabled, the Notification metadata nodes are present in the header. When the node 'enable-notification-envelope' is disabled, Notifications are encoded as defined in NETCONF Event Notifications [RFC5277]. Both extensions

defined in Section 3.2 and Section 3.3 are sent by default when (1) they are supported and (2) the node 'enable-notification-envelope' is enabled.

```
module: ietf-yp-notification

  augment /sn:subscriptions:
    +--rw enable-notification-envelope?   boolean
    +--rw metadata
```

When there are existing Subscriptions and a Client changes the node 'enable-notification-envelope', all existing Subscriptions MUST be terminated. The Publisher MUST send a 'subscription-terminated' notification to all the existing Subscriptions using the header configured before the change. Any new Subscription after the change uses the header defined by the node 'enable-notification-envelope', i.e. encoded as Section 3.1.1 when enabled and as defined in [RFC5277] if disabled.

The container 'metadata' is intended to support optional extensions that may be enabled on a per-extension basis. Defining explicit configuration knobs to enable or disable these extensions is optional and left to the extension designer. Refer to Appendix B for an example.

Per the YANG specifications (Section 7.17 of [RFC7950]), which does not allow making mandatory statements on augmentations, the 'enable-notification-envelope' YANG leaf must be made optional: the metadata fields MUST not be enabled when the Notification Envelope is not enabled.

Per-subscription support for this header was considered during the definition of this specification, however, a network node-wide setting was chosen to simplify network operations. In practice, operators use YANG-Push Subscriptions in large-scale deployments where numerous Subscriptions are created on each network node. While per-subscription configuration provides more flexibility, maintaining the choice of header format for each Subscription adds significant operational complexity when troubleshooting or migrating to a new header format. The desire for simplifying operations led to the definition of a global configuration knob for the Notification Envelope.

5. Discovering the Support of the Notification Envelope

A Client can discover the support of the Notification Envelope model through the capabilities model defined in [RFC9196]. This document extends the 'ietf-notification-capabilities' model with:

* A container containing a leaf 'envelope', stating that the YANG Notification can be encoded following the Notification Envelope model.

* A container 'metadata' containing all the supported extensions to this header. Extensions are defined in Section 3.2 and Section 3.3.

When the leafs defined in this document are supported by the Server, the Client discovers the presence of new metadata with the following augmentations in the 'ietf-notification-capabilities' module:

```
module: ietf-yp-notification

  augment /sysc:system-capabilities/notc:subscription-capabilities:
    +--ro notification-metadata
       +--ro envelope?   boolean
       +--ro metadata
          +--ro hostname-sequence-number?   boolean
                  {hostname-sequence-number}?

module: ietf-yp-observation

  augment /sysc:system-capabilities/notc:subscription-capabilities:
    +--ro yang-push-observation-time-supported?   boolean
```

6. Operational Considerations

As stated in Section 4, the Notification Envelope defined in this document replaces the header defined by NETCONF Event Notifications [RFC5277] when enabled. The NETCONF Event Notifications [RFC5277] header and the Notification Envelope defined in this document may coexist in the same network. Since enabling or disabling the Notification Envelope is performed on a YANG-Push Publisher basis, Receivers must only receive Notifications in a format they support. Operators should ensure that Receivers support the corresponding format or that proper validation and filtering are implemented when both formats are present in the same network.

In network devices where existing Subscriptions are active, enabling or disabling the Notification Envelope has operational impact. Operators should acknowledge that any existing Subscriptions will be terminated when enabling or disabling the Notification Envelope. Careful planning is therefore required to avoid unintended service disruption.

7. YANG Modules

7.1. The 'ietf-yp-notification' Module

The following sections show the YANG tree and YANG module for the 'ietf-yp-notification' module. This module defines the Notification Envelope structure, and extends the 'ietf-subscribed-notifications' and 'ietf-notification-capabilities' modules to enable both the configuration of the Notification Envelope and the discovery of supported extensions. This module supports the 'hostname' and 'sequence-number' as extensions to the Notification Envelope.

7.1.1. YANG 'ietf-yp-notification' Tree Diagram

This YANG module extends 'ietf-subscribed-notifications' [RFC8641] and 'ietf-notification-capabilities' [RFC9196] as shown in the following YANG tree [RFC8340]:

```
module: ietf-yp-notification

  augment /sn:subscriptions:
    +--rw enable-notification-envelope?   boolean
    +--rw metadata
  augment /sysc:system-capabilities/notc:subscription-capabilities:
    +--ro notification-metadata
       +--ro envelope?   boolean
       +--ro metadata
          +--ro hostname-sequence-number?   boolean
                  {hostname-sequence-number}?

  structure envelope:
    +-- event-time         yang:date-and-time
    +-- hostname?          inet:host-name {hostname-sequence-number}?
    +-- sequence-number?   yang:counter32 {hostname-sequence-number}?
    +-- contents?          <anydata>
```

7.1.2. YANG 'ietf-yp-notification' Module

The YANG module augments the module 'ietf-subscribed-notifications' [RFC8641], augments the module "ietf-notification-capabilities" [RFC9196] and uses 'ietf-yang-types' module [RFC6991] and 'ietf-yang-structure-ext' module [RFC8791].

```
<CODE BEGINS> file "ietf-yp-notification@2025-12-24.yang"
module ietf-yp-notification {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-yp-notification";
  prefix iypn;

  import ietf-yang-types {
```

```
      prefix yang;
      reference
        "draft-ietf-netmod-rfc6991-bis-18: Common YANG Data Types";
    }
    import ietf-inet-types {
      prefix inet;
      reference
        "draft-ietf-netmod-rfc6991-bis-18: Common YANG Data Types";
    }
    import ietf-subscribed-notifications {
      prefix sn;
      reference
        "RFC 8639: Subscription to YANG Notifications";
    }
    import ietf-system-capabilities {
      prefix sysc;
      reference
        "RFC 9196: YANG Modules Describing Capabilities for
         Systems and Datastore Update Notifications";
    }
    import ietf-notification-capabilities {
      prefix notc;
      reference
        "RFC 9196: YANG Modules Describing Capabilities for
         Systems and Datastore Update Notifications";
    }
    import ietf-yang-structure-ext {
      prefix sx;
      reference
        "RFC 8791: YANG Data Structure Extensions";
    }

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

       Authors:  Alex Huang Feng
                 <mailto:alex.huang-feng@insa-lyon.fr>
                 Pierre Francois
                 <mailto:pierre.francois@insa-lyon.fr>
                 Thomas Graf
                 <mailto:thomas.graf@swisscom.com>
                 Benoit Claise
                 <mailto:benoit@everything-ops.net>";
    description
      "Defines a notification header for Subscribed Notifications
```

```
      [RFC8639] and YANG-Push [RFC8641]. When this notification header
      is enabled through configuration, the root container of the
      notification is encoded as defined in RFCXXX.

      This module can be used to validate XML-encoded notifications
      [RFC7950], JSON-encoded messages [RFC7951], and CBOR-encoded
      messages [RFC9254]. Refer to Section 3.1.2 of RFC XXXX for more
      details.

      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).

      All revisions of IETF and IANA published modules can be found
      at the YANG Parameters registry group
      (https://www.iana.org/assignments/yang-parameters).

      This version of this YANG module is part of RFC XXXX; 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 2025-12-24 {
      description
        "Initial version.";
      reference
        "RFC XXXX: Extensible YANG Model for YANG-Push Notifications";
    }

    feature hostname-sequence-number {
      description
        "This feature indicates that hostname and sequence numbers are
        supported.";
    }

    grouping notif-env-capabilities {
      description
        "This grouping defines the capabilities for
         the notification-envelope defined in RFC XXXX
```

```
          and the different supported metadata.";
        leaf envelope {
          type boolean;
          default "true";
          description
            "Supports YANG-Push to use the notification-envelope as
             defined in RFC XXXX. If set to true, the publisher supports
             the notification envelope. If set to false, the
             notification envelope is not supported by the publisher.";
        }
        container metadata {
          description
            "Container with the supported optional metadata by the
             YANG-Push publisher.";
          leaf hostname-sequence-number {
            if-feature "hostname-sequence-number";
            type boolean;
            default "false";
            description
              "Supports hostname and sequence-number
               in the YANG-Push notifications as defined in the
               YANG-Push notification-envelope in RFC XXXX.
               If set to true, the publisher supports
               sending the hostname and sequence numbers
               within the notification envelope. If set to false,
               the hostname and sequence numbers are not supported.";
          }
        }
      }

      sx:structure envelope {
        leaf event-time {
          type yang:date-and-time;
          mandatory true;
          description
            "The date and time the event was generated by the network
             node.";
        }
        leaf hostname {
          if-feature "hostname-sequence-number";
          type inet:host-name;
          description
            "The hostname of the network node. This value is usually
             configured on the node by the administrator to identify
             the node in the network uniquely.";
        }
        leaf sequence-number {
          if-feature "hostname-sequence-number";
```

```
        type yang:counter32;
        description
          "Unique sequence number for each published message
           by the publisher process. The initial number is 1 and
           counts up by 1 at every published notification message
           until it reaches 4294967295. Then, it wraps around and
           restarts at 0. The value 0 is used to detect wrap
           arounds.";
      }
      anydata contents {
        description
          "This contains the values defined by the 'notification'
           statement unchanged.";
      }
    }

    // Subscription container
    augment "/sn:subscriptions" {
      description
        "This augmentation adds the configuration switches for
         enabling the notification envelope and metadata.";
      leaf enable-notification-envelope {
        type boolean;
        default "false";
        description
          "Enables YANG-Push to use the notification-envelope
          defined in RFC XXXX.

          Enabling or disabling this leaf terminates all
          existing active dynamic and configured YANG-Push
          subscriptions. The publisher MUST send a
          'subscription-terminated' notification to all the
          existing active subscriptions using
          the header configured before the change, then the
          subscription is terminated. Refer to
          Section 4 of RFC XXXX for more details.";
      }
      container metadata {
        description
          "Container for configuring optional metadata.
          Refer to Section 4 of RFC XXXX for more details.";
      }
    }

    // YANG-Push Capabilities extension
    augment "/sysc:system-capabilities"
          + "/notc:subscription-capabilities" {
      description
```

```
        "Extension to the subscription-capabilities model to enable
         clients to learn whether the publisher supports the
         notification-envelope";
      container notification-metadata {
        description
          "Adds the notification metadata capabilities to subscription
           capabilities.";
        uses notif-env-capabilities;
      }
    }
  }
  <CODE ENDS>
```

7.2. The 'ietf-yp-observation' Module

The following sections show the YANG tree and YANG module for the 'ietf-yp-observation' module. This module extends the 'ietf-yang-push' module to support sending the observation timestamp in 'push-update' and 'push-change-update' Notifications. Additionally, it augments the 'ietf-notification-capabilities' to enable Clients to discover the support of observation timestamps.

7.2.1. YANG 'ietf-yp-observation' Tree Diagram

This YANG module extends 'ietf-yang-push' [RFC8641] and 'ietf-notification-capabilities' [RFC9196] as shown in the following YANG tree [RFC8340]:

```
module: ietf-yp-observation

  augment /yp:push-update:
    +--ro timestamp?       yang:date-and-time
    +--ro point-in-time?   enumeration
  augment /yp:push-change-update:
    +--ro timestamp?       yang:date-and-time
    +--ro point-in-time?   enumeration
  augment /sysc:system-capabilities/notc:subscription-capabilities:
    +--ro yang-push-observation-time-supported?   boolean
```

7.2.2. YANG 'ietf-yp-observation' Module

The YANG module augments the module 'ietf-yang-push' [RFC8641], augments the module 'ietf-system-capabilities' [RFC9196].

```
   <CODE BEGINS> file "ietf-yp-observation@2025-12-24.yang"
   module ietf-yp-observation {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-yp-observation";
     prefix iypo;

     import ietf-yang-types {
       prefix yang;
       reference
         "RFC 6991: Common YANG Data Types";
     }
     import ietf-yang-push {
       prefix yp;
       reference
         "RFC 8641: Subscription to YANG Notifications for Datastore
         Updates";
     }
     import ietf-system-capabilities {
       prefix sysc;
       reference
         "RFC 9196: YANG Modules Describing Capabilities for
          Systems and Datastore Update Notifications";
     }
     import ietf-notification-capabilities {
       prefix notc;
       reference
         "RFC 9196: YANG Modules Describing Capabilities for
          Systems and Datastore Update Notifications";
     }

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

        Authors:  Thomas Graf
                  <mailto:thomas.graf@swisscom.com>
                  Benoit Claise
                  <mailto:benoit@everything-ops.net>
                  Alex Huang Feng
                  <mailto:alex.huang-feng@insa-lyon.fr>";
     description
       "Defines YANG-Push event notification header with the observation
       time in streaming update notifications.

       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).

      All revisions of IETF and IANA published modules can be found
      at the YANG Parameters registry group
      (https://www.iana.org/assignments/yang-parameters).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

    revision 2025-12-24 {
      description
        "Initial version.";
      reference
        "RFC XXXX: Extensible YANG Model for YANG-Push Notifications";
    }

    grouping yang-push-observation {
      description
        "This grouping adds the observation timestamp for the
        observed metrics.";
      leaf timestamp {
        type yang:date-and-time;
        description
          "This is the time when the metrics were observed.";
      }
      leaf point-in-time {
        type enumeration {
          enum current-accounting {
            description
              "For periodic subscriptions, the point-in-time
               where the metrics are being polled and observed.";
          }
          enum initial-state {
            description
              "For 'on-change sync on start' subscriptions, the
               initial point in time when the subscription was
               established and the state was observed.";
          }
          enum state-changed {
            description
              "For 'on-change sync on start' subscriptions, the
               point in time when the state change was observed after
               the subscription was established.";
          }
```

```
        }
        description
          "This describes at which point in time the metrics were
          observed.";
      }
    }

    // Event notifications
    augment "/yp:push-update" {
      description
        "This augmentation adds the observation timestamp of the
        accounted metrics in the push-update notification.";
      uses iypo:yang-push-observation;
    }

    augment "/yp:push-change-update" {
      description
        "This augmentation adds the observation timestamp of the
        event in the push-change-update notification.";
      uses iypo:yang-push-observation;
    }

    // Event capabilities
    augment "/sysc:system-capabilities"
          + "/notc:subscription-capabilities" {
      description
        "Add YANG-Push notification capabilities to system-level
        capability container.";
      leaf yang-push-observation-time-supported {
        type boolean;
        default "false";
        description
          "Specifies whether the publisher supports exporting
           observation-timestamp and point-in-time in notifications.
           If set to true, publisher supports. If set to false,
           the observation-timestamp is not supported.";
        reference
          "RFC XXXX: Extensible YANG Model for YANG-Push Notifications";
      }
    }
  }
  <CODE ENDS>
```

8. Implementation Status

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

8.1. Huawei VRP

Huawei implemented in push-update and push-change-update Notifications the timestamp and point-in-time extension as described in Section 3.3 for a YANG-Push Publisher on UDP-based Transport for Configured Subscriptions [I-D.ietf-netconf-udp-notif] in their VRP platform.

8.2. 6WIND VSR

6WIND implemented in push-update and push-change-update Notifications the timestamp and point-in-time extension as described in Section 3.3 for a YANG-Push Publisher on UDP-based Transport for Configured Subscriptions [I-D.ietf-netconf-udp-notif] in their VSR platform.

8.3. Cisco IOS XR

Cisco implemented in push-update and push-change-update Notifications the timestamp and point-in-time extension as described in Section 3.3 for a YANG-Push Publisher on UDP-based Transport for Configured Subscriptions [I-D.ietf-netconf-udp-notif] in their IOS XR platform.

9. Security Considerations

This section uses the template described in Section 3.7 of [I-D.ietf-netmod-rfc8407bis].

The 'ietf-yp-notification' and 'ietf-yp-observation' YANG modules define data models that are designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. These YANG-based management protocols (1) have to use a secure transport layer (e.g., SSH [RFC6242], TLS [RFC8446], and QUIC [RFC9000]) and (2) 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). All writable data nodes are likely to be reasonably sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. The following subtrees and data nodes have particular sensitivities/vulnerabilities:

* /sn:subscriptions/iypn:enable-notification-envelope

The entries in the list above will show whether the mechanism defined in this document is enabled. Access control MUST be set so that only someone with proper access permissions has the ability to access and modify this resource.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or Notification) to these data nodes. Specifically, the following subtrees and data nodes have particular sensitivities/ vulnerabilities:

* /iypn:envelope/hostname

The entries in the list above will show the identity of the originating device. Exposure of this information may assist an attacker in mapping the network or in injecting spoofed Notifications. Implementations SHOULD ensure that access to this data is restricted and that Notifications are sent over secure and authenticated channels.

10. IANA Considerations

This document describes the URI used for the IETF XML Registry and registers a new YANG module name.

10.1. URI

IANA is requested to add this document as a reference in the following URI's in the IETF XML Registry [RFC3688].

URI: urn:ietf:params:xml:ns:yang:ietf-yp-notification
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
Reference: RFC XXXX

URI: urn:ietf:params:xml:ns:yang:ietf-yp-observation
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.
Reference: RFC XXXX

10.2. YANG module name

This document registers the following YANG modules in the YANG Module Names Registry [RFC6020], within the "YANG Parameters" registry:

```
   name: ietf-yp-notification
   namespace: urn:ietf:params:xml:ns:yang:ietf-yp-notification
   prefix: iypn
   reference: RFC XXXX

   name: ietf-yp-observation
   namespace: urn:ietf:params:xml:ns:yang:ietf-yp-observation
   prefix: iypo
   reference: RFC XXXX
```

10.3. YANG SID-file

IANA is requested to register a new ".sid" file in the "IETF YANG SID Registry" [RFC9595]:

```
SID range entry point: TBD
SID range size: 50
YANG module name: ietf-yp-notification
reference: RFC XXXX
```

A ".sid" file is proposed in Appendix A.

Note to the RFC-Editor:

Please replace TBD with the value allocated by IANA.

11. Acknowledgements

The authors would like to thank Paul Aitken, Per Anderson, Andy Bierman, Carsten Bormann, Mohamed Boucadair, Tom Petch, Reshad Rahman, J端rgen Sch旦nw辰lder, Jason Sterne, Kent Watsen, Rob Wilton and Qin Wu for their review and valuable comments.

12. References

12.1. Normative References

[I-D.ietf-netmod-rfc6991-bis]
Sch旦nw辰lder, J., "Common YANG Data Types", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc6991-bis-18, 23 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc6991-bis-18>.

[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>.

[RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, DOI 10.17487/RFC5277, July 2008, <https://www.rfc-editor.org/info/rfc5277>.

[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>.

[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>.

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

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

[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>.

[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>.

[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>.

[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>.

[RFC8791] Bierman, A., Bj旦rklund, M., and K. Watsen, "YANG Data Structure Extensions", RFC 8791, DOI 10.17487/RFC8791, June 2020, <https://www.rfc-editor.org/info/rfc8791>.

[RFC9196] Lengyel, B., Clemm, A., and B. Claise, "YANG Modules Describing Capabilities for Systems and Datastore Update Notifications", RFC 9196, DOI 10.17487/RFC9196, February 2022, <https://www.rfc-editor.org/info/rfc9196>.

[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>.

[W3C.REC-xml-20001006] Bray, T., Paoli, J., Sperberg-McQueen, M., and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C, October 2000, <https://www.w3.org/TR/2000/REC-xml-20001006>.

12.2. Informative References

[I-D.ietf-netconf-udp-notif] Feng, A. H., Francois, P., Zhou, T., Graf, T., and P. Lucente, "UDP-based Transport for Configured Subscriptions", Work in Progress, Internet-Draft, draft-ietf-netconf-udp-notif-25, 28 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-udp-notif-25>.

[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>.

[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>.

[RFC6470] Bierman, A., "Network Configuration Protocol (NETCONF) Base Notifications", RFC 6470, DOI 10.17487/RFC6470, February 2012, <https://www.rfc-editor.org/info/rfc6470>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[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>.

[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>.

[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>.

[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>.

[RFC9232] Song, H., Qin, F., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Network Telemetry Framework", RFC 9232, DOI 10.17487/RFC9232, May 2022, <https://www.rfc-editor.org/info/rfc9232>.

[RFC9595] Veillette, M., Ed., Pelov, A., Ed., Petrov, I., Ed., Bormann, C., and M. Richardson, "YANG Schema Item iDentifier (YANG SID)", RFC 9595, DOI 10.17487/RFC9595, July 2024, <https://www.rfc-editor.org/info/rfc9595>.

Appendix A. .sid file

This appendix is not normative. For CBOR encoding using YANG-SIDs identifiers, a ".sid" file is requested to IANA in Section 10.3.

The .sid file can be found in TBD2.

Note to the RFC-Editor:

1. Please replace the entry-point and SID values with the ones allocated by IANA.

2. Please change the ’sid-file-status’ to ’published’ once the .sid file is published.

3. Please remove the .sid file from the document and insert a link in TBD2 to the .sid file reference from IANA.

<CODE BEGINS> file "ietf-yp-notification@2025-12-24.sid"


```
{
  "ietf-sid-file:sid-file": {
    "module-name": "ietf-yp-notification",
    "module-revision": "2025-12-24",
    "sid-file-status":"unpublished",
    "description": "draft-ietf-netconf-notif-envelope-04: Extensible YANG Model for YA
NG-Push Notifications",
    "dependency-revision": [
      {
        "module-name":"ietf-datastores",
        "module-revision":"2018-02-14"
      },
      {
        "module-name":"ietf-interfaces",
        "module-revision":"2018-02-20"
      },
      {
        "module-name":"ietf-ip",
        "module-revision":"2018-02-22"
      },
      {
        "module-name":"ietf-netconf-acm",
        "module-revision":"2018-02-14"
      },
      {
        "module-name":"ietf-network-instance",
        "module-revision":"2019-01-21"
      },
      {
        "module-name":"ietf-restconf",
        "module-revision":"2017-01-26"
      },
      {
        "module-name":"ietf-yang-library",
        "module-revision":"2019-01-04"
      },
      {
        "module-name":"ietf-yang-patch",
        "module-revision":"2017-02-22"
      },
      {
        "module-name":"ietf-yang-schema-mount",
        "module-revision":"2019-01-14"
      },
```

```
          {
            "module-name": "ietf-yang-types",
            "module-revision": "2025-06-23"
          },
          {
            "module-name": "ietf-inet-types",
            "module-revision": "2025-06-23"
          },
          {
            "module-name": "ietf-subscribed-notifications",
            "module-revision": "2019-09-09"
          },
          {
            "module-name": "ietf-system-capabilities",
            "module-revision": "2022-02-17"
          },
          {
            "module-name": "ietf-notification-capabilities",
            "module-revision": "2022-02-17"
          },
          {
            "module-name": "ietf-yang-structure-ext",
            "module-revision": "2020-06-17"
          }
        ],
        "assignment-range": [
          {
            "entry-point": "2950",
            "size": "50"
          }
        ],
        "item": [
          {
            "status":"unstable",
            "namespace":"module",
            "identifier":"ietf-yp-notification",
            "sid":"2950"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-subscribed-notifications:subscriptions/ietf-yp-notificatio
n:enable-notification-envelope",
            "sid":"2951"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-subscribed-notifications:subscriptions/ietf-yp-notificatio
n:metadata",
```

```
            "sid":"2952"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-system-capabilities:system-capabilities/ietf-notification-
capabilities:subscription-capabilities/ietf-yp-notification:notification-metadata",
            "sid":"2953"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-system-capabilities:system-capabilities/ietf-notification-
capabilities:subscription-capabilities/ietf-yp-notification:notification-metadata/envelop
e",
            "sid":"2954"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-system-capabilities:system-capabilities/ietf-notification-
capabilities:subscription-capabilities/ietf-yp-notification:notification-metadata/metadat
a",
            "sid":"2955"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-system-capabilities:system-capabilities/ietf-notification-
capabilities:subscription-capabilities/ietf-yp-notification:notification-metadata/metadat
a/hostname-sequence-number",
            "sid":"2956"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-yp-notification:envelope",
            "sid":"2957"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-yp-notification:envelope/contents",
            "sid":"2958"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-yp-notification:envelope/event-time",
            "sid":"2959"
          },
          {
            "status":"unstable",
            "namespace":"data",
            "identifier":"/ietf-yp-notification:envelope/hostname",
```

```
          "sid":"2960"
        },
        {
          "status":"unstable",
          "namespace":"data",
          "identifier":"/ietf-yp-notification:envelope/sequence-number",
          "sid":"2961"
        }
      ]
    }
  }
  <CODE ENDS>
```

Figure 11: ietf-yp-notification .sid file

Appendix B. Example extending the Notification Envelope

This appendix provides a non-normative example illustrating how an extension to the Notification Envelope should be implemented. The example does not define valid metadata, but serves to demonstrate how future extensions should be specified.

Specifically, the example extends the Notification Envelope with a metadata field named "foo". The YANG module illustrates the necessary augmentations to both the 'ietf-yp-notification' module and the 'ietf-notification-capabilities' module.

```
<CODE BEGINS> file "example-foo-extension.yang"
module example-foo-extension {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:example-foo-extension";
  prefix fooext;

  import ietf-subscribed-notifications {
    prefix sn;
  }
  import ietf-system-capabilities {
    prefix sysc;
  }
  import ietf-notification-capabilities {
    prefix notc;
  }
  import ietf-yang-structure-ext {
    prefix sx;
  }
  import ietf-yp-notification {
    prefix iypn;
  }
```

```
     description
       "Defines a new 'foo' metadata for the notification envelope.";

     // Extending the notification envelope header with a new
     // 'foo' metadata
     sx:augment-structure "/iypn:envelope" {
       leaf foo {
         type string;
         description
           "Description of the 'foo' extension.";
       }
     }

     // Extending the notifications capabilities so that clients can
     // learn whether this new extension is supported or not
     augment "/sysc:system-capabilities"
           + "/notc:subscription-capabilities"
           + "/iypn:notification-metadata/iypn:metadata" {
       description
         "Extension to the subscription-capabilities model to enable
          clients to learn whether the publisher supports the new
          'foo' metadata.";
       leaf foo {
         type boolean;
         default "false";
         description
           "Adds the 'foo' capability.";
       }
     }

     // (Optional) A user can optionally add knobs for enabling and
     // disabling specific metadata.
     augment "/sn:subscriptions/iypn:metadata" {
       description
         "An user can optionally, support a configuration knob for enabling
          or disabling a metadata.";
       leaf foo {
         type boolean;
         default "false";
         description
           "Configuration knob for enabling and disabling the 'foo'
            metadata.";
       }
     }
   }
   <CODE ENDS>
```

The extensions result in the following YANG tree:

```
   module: example-foo-extension

     augment /sysc:system-capabilities/notc:subscription-capabilities
               /iypn:notification-metadata/iypn:metadata:
       +--ro foo?   boolean
     augment /sn:subscriptions/iypn:metadata:
       +--rw foo?   boolean

     augment-structure /iypn:envelope:
       +-- foo?   string
```

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

Thomas Graf
Swisscom
Binzring 17
CH-8045 Zurich
Switzerland
Email: thomas.graf@swisscom.com

Benoit Claise
Everything OPS
Email: benoit@everything-ops.net