NETCONF
Internet-Draft
Intended status: Standards Track
Expires: 22 April 2022

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

19 October 2021

UDP-based Transport for Configured Subscriptions
draft-ietf-netconf-udp-notif-04

Abstract

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

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

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

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

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

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

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

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

2. Configured Subscription to UDP-Notif

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

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

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

3. UDP-Based Transport

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

3.1. Design Overview

As specified in Sub-Notif, the telemetry data is encapsulated in the NETCONF/RESTCONF notification message, which is then encapsulated and carried using transport protocols such as TLS or HTTP2. This document defines a UDP based transport. Figure 1 illustrates the structure of an UDP-Notif message.

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

* Notification Message is the encoded content that the publication stream transports. The common encoding methods include, CBOR [RFC7049], JSON, and XML. [I-D.ietf-netconf-notification-messages] describes the structure of the Notification Message for single notifications and bundled notifications.

```
                +-------+  +--------------+  +--------------+
                |  UDP  |  |   Message    |  | Notification |
                |       |  |   Header     |  | Message      |
                +-------+  +--------------+  +--------------+
```

Figure 1: UDP-Notif Message Overview

3.2. Format of the UDP-Notif Message Header

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

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-----+-+-------+---------------+-------------------------------+
     | Ver |S|  ET   |  Header Len   |         Message Length        |
     +-----+-+-------+---------------+-------------------------------+
     |                     Observation-Domain-ID                     |
     +---------------------------------------------------------------+
     |                          Message-ID                           |
     +---------------------------------------------------------------+
     ~                           Options                             ~
     +---------------------------------------------------------------+
```

Figure 2: UDP-Notif Message Header Format

The Message Header contains the following field:

* Ver represents the PDU (Protocol Data Unit) encoding version. The initial version value is 0.

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

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

  - 0: Reserved;

  - 1: JSON;

  - 2: XML;

  - 3: CBOR;

  - others are reserved.

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

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

* Observation-Domain-ID is a 32-bit identifier of the Observation Domain that led to the production of the notification message, as defined in [I-D.ietf-netconf-notification-messages]. This allows disambiguation of an information source, such as the identification of different line cards sending the notification messages. The source IP address of the UDP datagrams SHOULD NOT be interpreted as the identifier for the host that originated the UDP-Notif message. Indeed, the streamer sending the UDP-Notif message could be a relay for the actual source of data carried within UDP-Notif messages.

* The Message ID is generated continuously by the publisher of UDP-Notif messages. Different subscribers share the same Message ID sequence.

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

3.3. Data Encoding

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

Private encodings can be supported through the use of the S bit of the header. When the S bit is set, the value of the ET 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 ET field.

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

4. Options

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

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +---------------+---------------+--------------------------------
 |      Type     |     Length    |     Variable-length data
 +---------------+---------------+--------------------------------
```

Figure 3: Generic Option Format

* Type: 1 octet describing the option type;

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

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

When more than one option are used in the UDP-notif header, options MUST be ordered by the Type value.

4.1. Segmentation Option

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

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-----------------------------+-+
|     Type      |    Length     |        Segment Number       |L|
+---------------+---------------+-----------------------------+-+
```

Figure 4: Segmentation Option Format

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

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

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

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

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

An implementation of this specification MUST NOT rely on IP fragmentation by default to carry large messages. An implementation of this specification MUST either restrict the size of individual messages carried over this protocol, or support the segmentation option.

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

4.2. Private Encoding Option

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+--------------------------------
|     Type      |    Length     |  Variable length enc. descr.
+---------------+---------------+--------------------------------
```

Figure 5: Private Encoding Option Format

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

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

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

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

5. Applicability

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

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

5.1. Congestion Control

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

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

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

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

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

5.2. Message Size

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

5.3. Reliability

The target application for UDP-Notif is the collection of data-plane information. The lack of reliability of the data streaming mechanism is thus considered acceptable as the mechanism is to be used in controlled environments, mitigating the risk of information loss, while allowing for publication of very large amounts of data. Moreover, in this context, sporadic events when incomplete data collection is provided is not critical for the proper management of the network, as information collected for the devices through the means of the proposed mechanism is to be often refreshed.

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

5.4. Security Considerations

[RFC8085] states that "UDP applications that need to protect their communications againts 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 and thus, a security layer is unrequired. Nevertheless, a DTLS layer SHOULD be implemented in open or unsecured networks. A DTLS layered implementation is presented in Section 6.

6. Secured layer for UDP-notif

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

The DTLS 1.3 protocol [I-D.draft-ietf-tls-dtls13] is designed to meet the requirements of applications that need to secure datagram transport.

DTLS can be used as a secure transport to counter all the primary threats to UDP-notif:

* Confidentiality to counter disclosure of the message contents.

* Integrity checking to counter modifications to a message on a hop-by-hop basis.

* Server or mutual authentication to counter masquerade.

In addition, DTLS also provides:

* A cookie exchange mechanism during handshake to counter Denial of Service attacks.

* A sequence number in the header to counter replay attacks.

Even though this security layer is unrequired, DTLS 1.3 SHOULD be implemented on unsecured networks to achieve privacy.

6.1. Transport

As shown in Figure 6, the DTLS is layered next to the UDP transport providing reusable security and authentication functions over UDP. No DTLS extension is required to enable UDP-notif messages over DTLS.

```
+------------------------------+
|      UDP-notif Message       |
+------------------------------+
|             DTLS             |
+------------------------------+
|             UDP              |
+------------------------------+
|             IP               |
+------------------------------+
```

Figure 6: Protocol Stack for DTLS secured UDP-notif

The application implementer will map a unique combination of the remote address, remote port number, local address, and local port number to a session.

Each UDP-notif message is delivered by the DTLS record protocol, which assigns a sequence number to each DTLS record. Although the DTLS implementer may adopt a queue mechanism to resolve reordering, it may not assure that all the messages are delivered in order when mapping on the UDP transport.

Since UDP is an unreliable transport, with DTLS, an originator or a relay may not realize that a collector has gone down or lost its DTLS connection state, so messages may be lost.

The DTLS record has its own sequence number, encryption and decryption will be done by the DTLS layer, so that the UDP-notif Message layer is not impacted by the use of DTLS.

6.2. Port Assignment

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

6.3. Session lifecycle

6.3.1. DTLS Session Initiation

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

When DTLS is implemented, the Publisher MUST NOT send any UDP-notif messages before the DTLS handshake has successfully completed.

Implementations of this security layer MUST support DTLS 1.3 [I-D.draft-ietf-tls-dtls13] and MUST support the mandatory to implement cipher suite TLS_AES_128_GCM_SHA256 and SHOULD implement TLS_AES_256_GCM_SHA384 and TLS_CHACHA20_POLY1305_SHA256 cipher suites, as specified in TLS 1.3 [RFC8446]. If additional cipher suites are supported, then implementations MUST NOT negotiate a cipher suite that employs NULL integrity or authentication algorithms.

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

6.3.2. 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.

6.3.3. Session termination

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

When no data is received from a DTLS connection for a long time, the Receiver MAY close the connection. Implementations SHOULD set the timeout value to 10 minutes but application specific profiles MAY recommend shorter or longer values. The Receiver (DTLS server) MUST attempt to initiate an exchange of close_notify alerts with the Publisher before closing the connection. Receivers that are unprepared to receive any more data MAY close the connection after sending the close_notify alert.

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

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

The YANG model defined in Section 8 has two leaves augmented into one place of Sub-Notif [RFC8639], plus one identity.

```
    module: ietf-udp-subscribed-notifications
     augment /sn:subscriptions/sn:subscription/sn:receivers/sn:receiver:
       +--rw address   inet:ip-address
       +--rw port      inet:port-number
       +--rw enable-segmentation?  boolean
       +--rw max-segmentation-size?  uint32
```

8. YANG Module

```
   <CODE BEGINS> file "ietf-udp-notif@2020-10-18.yang"
   module ietf-udp-notif {
     yang-version 1.1;
     namespace
       "urn:ietf:params:xml:ns:yang:ietf-udp-notif";
     prefix un;
     import ietf-subscribed-notifications {
       prefix sn;
       reference
         "RFC 8639: Subscription to YANG Notifications";
     }
     import ietf-inet-types {
       prefix inet;
       reference
         "RFC 6991: Common YANG Data Types";
     }

     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>
                  Paolo Lucente
                  <mailto:paolo@ntt.net>";

     description
       "Defines UDP-Notif as a supported transport for subscribed
       event notifications.

       Copyright (c) 2018 IETF Trust and the persons identified as authors
       of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or without
       modification, is permitted pursuant to, and subject to the license
       terms contained in, the Simplified BSD License set forth in Section
       4.c of the IETF Trust's Legal Provisions Relating to IETF Documents
       (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 2021-10-18 {
       description
       "Slight change to the name of two parameters.";
       reference
       "RFC XXXX: UDP-based Transport for Configured Subscriptions";
     }


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

     identity encode-cbor {
       base sn:encoding;
       description
         "Encode data using CBOR as described in RFC 7049.";
       reference
         "RFC 7049: Concise Binary Object Representation";
     }

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

       leaf address {
         type inet:ip-address;
         mandatory true;
         description
           "IP address of target UDP-Notif receiver, which can be an
           IPv4 address or an IPV6 address.";
       }

       leaf port {
       type inet:port-number;
       description
         "Port number of target UDP-Notif receiver, if not specified,
         the system should use default port number.";
       }

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

       leaf max-segmentation-size {
       when "../enable-segmentation = 'true'";
       type uint32;
       description "UDP-Notif provides a configurable
         max-segmentation-size to control the size of each message.";
       }
     }

     augment "/sn:subscriptions/sn:subscription/sn:receivers/sn:receiver" {
       when "derived-from(../../../transport, 'un:udp-notif')";
       description
         "This augmentation allows UDP-Notif specific parameters to be
          exposed for a subscription.";

       uses target-receiver;
     }
   }
   <CODE ENDS>
```

9. IANA Considerations

This document is creating 2 registries called "UDP-notif encoding types" and "UDP-notif option types" under the new heading "UDP-notif protocol". The registration procedure is made using the Standards Action process defined in [RFC8126].

The first requested registry is the following:

Registry Name: UDP-notif encoding 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 encoding types":

Value: 0
Description: Reserved
Reference: this document

Value: 1
Description: Payload encoded in JSON
Reference: this document

Value: 2
Description: Payload encoded in XML
Reference: this document

Value: 3
Description: Payload encoded in CBOR
Reference: this document

The second requested registry is the following:

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

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

Value: 0
Description: Reserved
Reference: this document

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

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

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

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

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

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

10. Acknowledgements

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

11. References

11.1. Normative References

[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>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

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

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

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

[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>.

11.2. Informative References

[I-D.draft-ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, July 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-dtls13-43>.

[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-02, May 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-distributed-notif-02>.

[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-08, 22 February 2021, <https://www.ietf.org/archive/id/draft-ietf-netconf-https-notif-08.txt>.

[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://www.ietf.org/archive/id/draft-ietf-netconf-notification-messages-08.txt>.

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

NETCONF J. Lindblad
Internet-Draft Cisco Systems
Intended status: Standards Track 22 October 2021
Expires: 25 April 2022

Transaction ID Mechanism for NETCONF
draft-lindblad-netconf-transaction-id-01

Abstract

NETCONF clients and servers often need to have a synchronized view of the server's configuration data stores. The volume of configuration data in a server may be very large, while data store changes typically are small when observed at typical client resynchronization intervals.

Rereading the entire data store and analyzing the response for changes is an inefficient mechanism for synchronization. This document specifies an extension to NETCONF that allows clients and servers to keep synchronized with a much smaller data exchange and without any need for servers to store information about the clients.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/janlindblad/netconf-transaction-id.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

When a NETCONF client connects with a NETCONF server, a frequently occurring use case is for the client to find out if the configuration has changed since it was last connected. Such changes could occur for example if another NETCONF client has made changes, or another system or operator made changes through other means than NETCONF.

One way of detecting a change for a client would be to retrieve the entire configuration from the server, then compare the result with a previously stored copy at the client side. This approach is not popular with most NETCONF users, however, since it would often be very expensive in terms of communications and computation cost.

Furthermore, even if the configuration is reported to be unchanged, that will not guarantee that the configuration remains unchanged when a client sends a subsequent change request, a few moments later.

Evidence of a transaction id feature being demanded by clients is that several server implementors have built proprietary and mutually incompatible mechanisms for obtaining a transaction id from a NETCONF server.

RESTCONF, RFC 8040 (https://tools.ietf.org/html/rfc8040), defines a mechanism for detecting changes in configuration subtrees based on Entity-tags (ETags). In conjunction with this, RESTCONF provides a way to make configuration changes conditional on the server confiuguration being untouched by others. This mechanism leverages RFC 7232 (https://tools.ietf.org/html/rfc7232) "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests".

This document defines similar functionality for NETCONF, RFC 6241 (https://tools.ietf.org/html/rfc6241).

2. Conventions and Definitions

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.

3. NETCONF Transaction id Extension

This document describes a NETCONF extension which modifies the behavior of get-config, get-data, edit-config, edit-data, discard-changes, copy-config, delete-config and commit such that clients are able to conditionally retrieve and update the configuration in a NETCONF server. NETCONF servers that support this extension MUST announce the capability "urn:ietf:params:netconf:capability:txid:1.0".

Several low level mechanisms could be defined to fulfill the requirements for efficient client-server transaction id synchronization. This document defines only one mechanism, but additional mechanisms could be added in future versions of this document, or in separate documents.

The common use cases for such mecahnisms are briefly discussed here.

Initial configuration retrieval When the client initially connects to a server, it may be interested to acquire a current view of (parts of) the server’s configuration.
In order to be able to efficiently detect changes later, it may also be interested to store meta level transaction id information about subtrees of the configuration.

Subsequent configuration retrieval When a client needs to reread (parts of) the server’s configuration, it may be interested to leverage the transaction id meta data it has stored by requesting the server to prune the response so that it does not repeat configuration data that the client is already aware of.

Configuration update with transaction id return When a client issues a transaction towards a server, it may be interested to also learn the new transaction id meta data the server has stored for the updated parts of the configuration.

Configuration update with transaction id specification When a client issues a transaction towards a server, it may be interested to also specify the new transaction id meta data that the server stores for the updated parts of the configuration.

Conditional configuration update When a client issues a transaction towards a server, it may specify transaction id data for the transaction in order to allow the server to verify that the client is up to date with any changes in the parts of the configuration that it is concerned with. If the transaction id information in the server is different than the client expected, the server rejects the transaction with a specific error message.

3.1. General Principles

All transaction id mechanisms SHALL maintain a transaction id value for each configuration datastore supported by the server. Some transaction id mechanisms will also maintain transaction id values for elements deeper in the YANG data tree. The elements for which the server maintains transaction ids are collectively referred to as the "versioned elements".

The server returning transaction id values for the versioned elements MUST ensure the transaction id values are changed every time there has been a configuration change at or below the element associated with the value. This means any update of a config true element will result in a new transaction id value for all ancestor versioned elements, up to and including the datastore root itself.

This also means a server MUST update the transaction id value for any elements that change as a result of a configuration change, regardless of source, even if the changed elements are not explicitly part of the change payload. An example of this is dependent data under YANG RFC 7950 (https://tools.ietf.org/html/rfc7950) when- or choice-statements.

The server MUST NOT change the transaction id value of a versioned element unless a child element of that element has been changed. The server MUST NOT change any transaction id values due to changes in config false data.

3.2. Conditional Transactions

Conditional transactions are useful when a client is interested to make a configuration change, being sure that the server configuration has not changed since the client last inspected it.

By supplying the latest transaction id values known to the client in its change requests (edit-config etc.), it can request the server to reject the transaction in case any relevant changes have occurred at the server that the client is not yet aware of.

This allows a client to reliably compute and send confiuguration changes to a server without either acquiring a global datastore lock for a potentially extended period of time, or risk that a change from another client disrupts the intent in the time window between a read (get-config etc.) and write (edit-config etc.) operation.

If the server rejects the transaction because the configuration transaction id value differs from the client's expectation, the server MUST return an rpc-error with the following values:

```
error-tag:      operation-failed
error-type:     protocol
error-severity: error
```

Additionally, the error-info tag SHOULD contain an sx:structure containing relevant details about the mismatching transaction ids.

3.3. Other NETCONF Operations

discard-changes The discard-changes operation resets the candidate datastore to the contents of the running datastore. The server MUST ensure the transaction id values in the candidate datastore get the same values as in the running datastore when this operation runs.

copy-config The copy-config operation can be used to copy contents between datastores. The server MUST ensure the transaction id values retain the same values as in the soruce datastore.

If copy-config is used to copy from a file, URL or other source that is not a datastore, the server MUST ensure the transaction id values are changed.

delete-config The server MUST ensure the datastore transaction id value is changed.

commit At commit, with regards to the transaction id values, the server MUST treat the contents of the candidate datastore as if any transaction id value provided by the client when updating the candidate was provided in a single edit-config towards the running datastore. If the transaction is rejected due to transaction id value mismatch, an rpc-error as described in section Conditional Transactions (Section 3.2) MUST be sent.

4. ETag Transaction id Mechanism

4.1. ETag attribute

Central to the ETag configuration retrieval and update mechanism described in the following sections is a meta data XML attribute called "etag". The etag attribute is defined in the namespace "urn:ietf:params:xml:ns:netconf:txid:1.0".

Servers MUST maintain a top-level etag value for each configuration datastore they implement. Servers SHOULD maintain etag values for YANG containers that hold configuration for different subsystems. Servers MAY maintain etag values for any YANG container or list element they implement.

The etag attribute values are opaque UTF-8 strings chosen freely, except that the etag string must not contain space, backslash or double quotes. The point of this restriction is to make it easy to reuse implementations that adhere to section 2.3.1 in RFC 7232 (https://tools.ietf.org/html/rfc7232). The probability SHOULD be made very low that an etag value that has been used historically by a server is used again by that server.

The detailed rules for when to update the etag value are described in section Configuration Update (Section 4.3). These rules are chosen to be consistent with the ETag mechanism in RESTCONF, RFC 8040 (https://tools.ietf.org/html/rfc8040), specifically sections 3.4.1.2, 3.4.1.3 and 3.5.2.

4.2. Configuration Retreival

Clients MAY request the server to return etag attribute values in the response by adding one or more etag attributes in get-config or get-data requests.

The etag attribute may be added directly on the get-config or get-data requests, in which case it pertains to the entire datastore. A client MAY also add etag attributes to zero or more individual elements in the get-config or get-data filter, in which case it pertains to the subtree rooted at that element.

For each element that the client requests etag attributes, the server MUST return etags for all versioned elements at or below that point that are part of the server's respone. ETags are returned as attributes on the element they pertain to. The datastore root etag value is returned on the top-level data tag in the response.

If the client is requesting an etag value for an element that is not among the server's versioned elements, then the server MUST return the etag attribute on the closest ancestor that is a versioned element, and all children of that ancestor. The datastore root is always a versioned element.

4.2.1. Initial Configuration Response

When the client adds etag attributes to a get-config or get-data request, it should specify the last known etag values it has seen for the elements it is asking about. Initially, the client will not know any etag value and should use "?".

To retrieve etag attributes across the entire NETCONF server configuration, a client might send:

```
   <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
        xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
     <get-config txid:etag="?"/>
   </rpc>
```

To retrieve etag attributes for a specific interface using an xpath filter, a client might send:

```
   <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
        xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
     <get-config>
       <source>
         <running/>
       </source>
       <filter type="xpath"
         xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
         select=
           "/if:interfaces/if:interface[if:name='GigabitEthernet-0/0']"
         txid:etag="?"/>
     </get-config>
   </rpc>
```

To retrieve etag attributes for "ietf-interfaces", but not for "nacm", a client might send:

```
   <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
        xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
     <get-config>
       <source>
         <running/>
       </source>
       <filter>
         <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
           txid:etag="?"/>
         <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
       </filter>
     </get-config>
   </rpc>
```

When a NETCONF server receives a get-config or get-data request containing txid:etag attributes with the value "?", it MUST return etag attributes for all versioned elements below this point included in the reply.

If the server considers the container "interfaces" and the list "interface" elements to be versioned elements, the server's response to the request above might look like:

```
   <rpc-reply message-id="1"
              xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
              xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
     <data txid:etag="def88884321">
       <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                   txid:etag="def88884321">
         <interface txid:etag="def88884321">
           <name>GigabitEthernet-0/0</name>
           <description>Management Interface</description>
           <type>ianaift:ethernetCsmacd</type>
           <enabled>true</enabled>
         </interface>
         <interface txid:etag="abc12345678">
           <name>GigabitEthernet-0/1</name>
           <description>Upward Interface</description>
           <type>ianaift:ethernetCsmacd</type>
           <enabled>true</enabled>
         </interface>
       </interfaces>
       <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
         <groups>
           <group>
             <name>admin</name>
             <user-name>sakura</user-name>
             <user-name>joe</user-name>
           </group>
         </groups>
       </nacm>
     </data>
   </rpc>
```

4.2.2. Configuration Response Pruning

A NETCONF client that already knows some etag values MAY request that the configuration retrieval request is pruned with respect to the client’s prior knowledge.

To retrieve only changes for "ietf-interfaces" that do not have the last known etag value "abc12345678", but include the entire configuration for "nacm", regardless of etags, a client might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        txid:etag="abc12345678"/>
      <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
    </filter>
  </get-config>
</rpc>
```

When a NETCONF server receives a get-config or get-data request containing an element with a client specified etag attribute, there are several different cases:

* The element is not a versioned element, i.e. the server does not maintain an etag value for this element. In this case, the server MUST look up the closest ancestor that is a versioned element, and proceed as if the client had specified the etag value for that element.

* The element is a versioned element, and the client specified etag attribute value is different than the server's etag value for this element. In this case the server MUST return the contents as it would otherwise have done, adding the etag attributes of all child versioned elements to the response. In case the client has specified etag attributes for some child elements, then these cases MUST be re-evaluated for those elements.

* The element is a versioned element, and the client specified etag attribute value matches the server's etag value. In this case the server MUST return the element decorated with an etag attribute with the value "=", and child elements pruned.

For list elements, pruning child elements means that key elements MUST be included in the response, and other child elements MUST NOT be included. For containers, child elements MUST NOT be included.

For example, assuming the NETCONF server configuration is the same as in the previous rpc-reply example, the server's response to request above might look like:

```
   <rpc-reply message-id="1"
              xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
              xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
     <data txid:etag="def88884321">
       <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                   txid:etag="def88884321">
         <interface txid:etag="def88884321">
           <name>GigabitEthernet-0/0</name>
           <description>Management Interface</description>
           <type>ianaift:ethernetCsmacd</type>
           <enabled>true</enabled>
         </interface>
         <interface txid:etag="=">
           <name>GigabitEthernet-0/1</name>
         </interface>
       </interfaces>
       <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"/>
         <groups>
           <group>
             <name>admin</name>
             <user-name>sakura</user-name>
             <user-name>joe</user-name>
           </group>
         </groups>
       </nacm>
     </data>
   </rpc>
```

4.3. Configuration Update

Whenever the configuration on a server changes for any reason, the server MUST update the etag value for all versioned elements that have children that changed.

If the change is due to a NETCONF client edit-config or edit-data request that includes the ietf-netconf-txid:with-etag presence container, the server MUST return the etag value of the targeted datastore as an attribute on the XML ok tag in the rpc-reply.

The server MUST NOT change the etag value of a versioned element unless a child element of that element has been changed. The server MUST NOT change any etag values due to changes in config false data.

How the server selects a new etag value to use for the changed elements is described in section ETag attribute (Section 4.1).

For example, if a client wishes to update the interface description for interface "GigabitEthernet-0/1" to "Downward Interface", it might send:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
               xmlns:ietf-netconf-txid=
                "urn:ietf:params:xml:ns:yang:ietf-netconf-txid">
    <target>
      <candidate/>
    </target>
    <test-option>test-then-set</test-option>
    <ietf-netconf-txid:with-etag/>
    <config>
      <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
        <interface>
          <name>GigabitEthernet-0/1</name>
          <description>Downward Interface</description>
        </interface>
      </interfaces>
    </config>
  </edit-config>
</rpc>
```

The server would update the description leaf in the candidate datastore, and return an rpc-reply as follows:

```
<rpc-reply message-id="1"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <ok txid:etag="ghi55550101"/>
</rpc-reply>
```

A subsequent get-config request for "ietf-interfaces", with txid:etag="?" might then return:

```
   <rpc-reply message-id="1"
              xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
              xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
     <data txid:etag="ghi55550101">
       <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                   txid:etag="ghi55550101">
         <interface txid:etag="def88884321">
           <name>GigabitEthernet-0/0</name>
           <description>Management Interface</description>
           <type>ianaift:ethernetCsmacd</type>
           <enabled>true</enabled>
         </interface>
         <interface txid:etag="ghi55550101">
           <name>GigabitEthernet-0/1</name>
           <description>Downward Interface</description>
           <type>ianaift:ethernetCsmacd</type>
           <enabled>true</enabled>
         </interface>
       </interfaces>
     </data>
   </rpc>
```

In case the server at this point received a configuration change from another source, such as a CLI operator, adding an MTU value for the interface "GigabitEthernet-0/0", a subsequent get-config request for "ietf-interfaces", with txid:etag="?" might then return:

```
   <rpc-reply message-id="1"
              xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
              xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
     <data txid:etag="cli22223333">
       <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                   txid:etag="cli22223333">
         <interface txid:etag="cli22223333">
           <name>GigabitEthernet-0/0</name>
           <description>Management Interface</description>
           <type>ianaift:ethernetCsmacd</type>
           <enabled>true</enabled>
           <mtu>768</mtu>
         </interface>
         <interface txid:etag="ghi55550101">
           <name>GigabitEthernet-0/1</name>
           <description>Downward Interface</description>
           <type>ianaift:ethernetCsmacd</type>
           <enabled>true</enabled>
         </interface>
       </interfaces>
     </data>
   </rpc>
```

4.3.1. Conditional Configuration Update

When a NETCONF client sends an edit-config or edit-data request to a NETCONF server that implements this specification, the client MAY specify expected etag values on the versioned elements touched by the transaction.

If such an etag value differs from the etag value stored on the server, the server MUST reject the transaction and return an rpc-error as specified in section Conditional Transactions (Section 3.2).

Additionally, the error-info tag MUST contain an sx:structure etag-value-mismatch-error-info as defined in the module ietf-netconf-txid, with mismatch-path set to the instance identifier value identifying one of the versioned elements that had an etag value mismatch, and mismatch-etag-value set to the server's current value of the etag attribute for that versioned element.

For example, if a client wishes to delete the interface "GigabitEthernet-0/1" if and only if its configuration has not been altered since this client last synchronized its configuration with the server (at which point it received the etag "ghi55550101"), regardless of any possible changes to other interfaces, it might send:

```
   <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
        xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
        xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0"
        xmlns:ietf-netconf-txid=
          "urn:ietf:params:xml:ns:yang:ietf-netconf-txid">
     <edit-config>
       <target>
         <candidate/>
       </target>
       <test-option>test-then-set</test-option>
       <ietf-netconf-txid:with-etag/>
       <config>
         <interfaces
           xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
           <interface nc:operation="delete"
                      txid:etag="ghi55550101">
             <name>GigabitEthernet-0/1</name>
           </interface>
         </interfaces>
       </config>
     </edit-config>
   </rpc>
```

If interface "GigabitEthernet-0/1" has the etag value "ghi55550101", as expected by the client, the transaction goes through, and the server responds something like:

```
   <rpc-reply message-id="1"
              xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
              xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
     <ok txid:etag="xyz77775511"/>
   </rpc-reply>
```

A subsequent get-config request for "ietf-interfaces", with txid:etag="?" might then return:

```
   <rpc-reply message-id="1"
              xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
              xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
     <data txid:etag="xyz77775511">
       <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
                   txid:etag="xyz77775511">
         <interface txid:etag="def88884321">
           <name>GigabitEthernet-0/0</name>
           <description>Management Interface</description>
           <type>ianaift:ethernetCsmacd</type>
           <enabled>true</enabled>
         </interface>
       </interfaces>
     </data>
   </rpc-reply>
```

In case interface "GigabitEthernet-0/1" did not have the expected etag value "ghi55550101", the server rejects the transaction, and might send:

```
   <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
              xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
              xmlns:ietf-netconf-txid=
                "urn:ietf:params:xml:ns:yang:ietf-netconf-txid">
              message-id="1">
     <rpc-error>
       <error-type>protocol</error-type>
       <error-tag>operation-failed</error-tag>
       <error-severity>error</error-severity>
       <error-info>
         <ietf-netconf-txid:etag-value-mismatch-error-info>
           <ietf-netconf-txid:mismatch-path>
             /if:interfaces/if:interface[if:name="GigabitEthernet-0/0"]
           </ietf-netconf-txid:mismatch-path>
           <ietf-netconf-txid:mismatch-etag-value>
             cli22223333
           </ietf-netconf-txid:mismatch-etag-value>
         </ietf-netconf-txid:etag-value-mismatch-error-info>
       </error-info>
     </rpc-error>
   </rpc-reply>
```

4.4. ETags with Other NETCONF Operations

The following NETCONF Operations also need some special considerations.

discard-changes The server MUST ensure the etag attributes in the candidate datastore get the same values as in the running datastore when this operation runs.

copy-config The server MUST ensure the etag attributes retain the same values as in the soruce datastore.

If copy-config is used to copy from a source that is not a datastore, the server MUST ensure etags are given new values.

delete-config The server MUST ensure the datastore etag is given a new value.

commit At commit, with regards to the etag values, the server MUST treat the contents of the candidate datastore as if any etag attributes provided by the client were provided in a single edit-config towards the running datastore. If the commit is rejected due to etag mismatch, the rpc-error message specified in section Conditional Configuration Update (Section 4.3.1) MUST be sent.

The client MAY request that the new etag value is returned as an attribute on the ok response for a successful commit. The client requests this by adding with-etag to the commit operation.

For example, a client might send:

```
<rpc message-id="1"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    xmlns:ietf-netconf-txid=
      "urn:ietf:params:xml:ns:yang:ietf-netconf-txid"
  <commit>
    <ietf-netconf-txid:with-etag/>
  </commit>
</rpc>
```

Assuming the server accepted the transaction, it might respond:

```
<rpc-reply message-id="1"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:txid="urn:ietf:params:xml:ns:netconf:txid:1.0">
  <ok txid:etag="ghi55550101"/>
</rpc-reply>
```

5. YANG Modules

```
   module ietf-netconf-txid {
     yang-version 1.1;
     namespace
       'urn:ietf:params:xml:ns:yang:ietf-netconf-txid';
     prefix ietf-netconf-txid;

     import ietf-netconf {
       prefix nc;
     }

     import ietf-netconf-nmda {
       prefix ncds;
     }

     import ietf-yang-structure-ext {
       prefix sx;
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group";

     contact
       "WG Web:   <http://tools.ietf.org/wg/netconf/>
        WG List:  <netconf@ietf.org>

        Author:   Jan Lindblad
                  <mailto:jlindbla@cisco.com>";

     description
       "NETCONF Transaction ID aware operations for NMDA.

        Copyright (c) 2021 IETF Trust and the persons identified as
        the document authors.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

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

     revision 2021-11-01 {
       description
         "Initial revision";
       reference
```

```
         "RFC XXXX: Xxxxxxxxx";
     }

     typedef etag-t {
       type string {
         pattern ".* .*" {
           modifier invert-match;
         }
         pattern ".*\".*" {
           modifier invert-match;
         }
         pattern ".*\\.*" {
           modifier invert-match;
         }
       }
       description
         "Unique Entity-tag value representing a specific transaction.
          Could be any string that does not contain spaces, double
          quotes or backslash.  The values '?' and '=' have special
          meaning.";
     }

     grouping transaction-id-grouping {
       container with-etag {
         presence
           "Indicates that the client requests the server to include a
            txid:etag transaction id in the rpc-reply";
       }
       description
         "Grouping for transaction id mechanisms, to be augmented into
          rpcs that modify configuration data stores.";
     }

     augment /nc:edit-config/nc:input {
       uses transaction-id-grouping;
       description
         "Injects the transaction id mechanisms into the
         edit-config operation";
     }

     augment /nc:commit/nc:input {
       uses transaction-id-grouping;
       description
         "Injects the transaction id mechanisms into the
         commit operation";
     }

     augment /ncds:edit-data/ncds:input {
       uses transaction-id-grouping;
       description
         "Injects the transaction id mechanisms into the
         edit-data operation";

     sx:structure etag-value-mismatch-error-info {
       container etag-value-mismatch-error-info {
         description
            "This error is returned by a NETCONF server when a client
             sends a configuration change request, with the additonal
             condition that the server aborts the transaction if the
             server's configuration has changed from what the client
             expects, and the configuration is found not to actually
             not match the client's expectation.";
         leaf mismatch-path {
           type instance-identifier;
           description
             "Indicates the YANG path to the element with a mismatching
              etag value.";
         }
         leaf mismatch-etag-value {
           type etag-t;
           description
             "Indicates server's value of the etag attribute for one
              mismatching element.";
         }
       }
     }
   }
```

6. Security Considerations

TODO Security

7. IANA Considerations

This document registers the following capability identifier URN in the 'Network Configuration Protocol (NETCONF) Capability URNs' registry:

```
     urn:ietf:params:netconf:capability:txid:1.0
```

This document registers two XML namespace URNs in the 'IETF XML registry', following the format defined in RFC 3688 (https://tools.ietf.org/html/rfc3688).

URI: urn:ietf:params:xml:ns:netconf:txid:1.0

URI: urn:ietf:params:xml:ns:yang:ietf-netconf-txid

Registrant Contact: The NETCONF WG of the IETF.

XML: N/A, the requested URIs are XML namespaces.

This document registers one module name in the 'YANG Module Names' registry, defined in RFC 6020 (https://tools.ietf.org/html/rfc6020).

name: ietf-netconf-txid

prefix: ietf-netconf-txid

namespace: urn:ietf:params:xml:ns:yang:ietf-netconf-txid

RFC: XXXX

8. Changes

8.1. Major changes in -01 since -00

* Updated the text on numerous points in order to answer questions that appeared on the mailing list.

* Changed the document structure into a general transaction id part and one etag specific part.

* Renamed entag attribute to etag, prefix to txid, namespace to urn:ietf:params:xml:ns:yang:ietf-netconf-txid.

* Set capability string to urn:ietf:params:netconf:capability:txid:1.0

* Changed YANG module name, namespace and prefix to match names above.

* Harmonized/slightly adjusted etag value space with RFC 7232 and RFC 8040.

* Removed all text discussing etag values provided by the client (although this is still an interesting idea, if you ask the author)

* Clarified the etag attribute mechanism, especially when it comes to matching against non-versioned elements, its cascading upwards in the tree and secondary effects from when- and choice-statements.

* Added a mechanism for returning the server assigned etag value in get-config and get-data.

* Added section describing how the NETCONF discard-changes, copy-config, delete-config and commit operations work with respect to etags.

* Added IANA Considerations section.

* Removed all comments about open questions.

9. Normative References

[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/rfc/rfc2119>.

[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/rfc/rfc8174>.

Acknowledgments

The author wishes to thank Benoit Claise for making this work happen, and the following individuals, who all provided helpful comments: Per Andersson, Kent Watsen, Andy Bierman, Robert Wilton, Qiufang Ma.

Author's Address

Jan Lindblad
Cisco Systems

Email: jlindbla@cisco.com

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 25 April 2022

Q. Wu
W. Song
Huawei
P. Liu
China Mobile
Q. Ma
Huawei
W. Wang
China Telecom

22 October 2021

Adaptive Subscription to YANG Notification
draft-wang-netconf-adaptive-subscription-07

Abstract

This document defines a YANG data model and associated mechanism enabling subscriber's adaptive subscriptions to a publisher's event streams with various different period intervals to report updates. Applying these elements allows servers automatically adjust the volume of telemetry traffic and rate of traffic sent from publisher to the receivers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

YANG-Push subscriptions [RFC8641] allow client applications to subscribe to continuous datastore updates without needing to poll. It defines a mechanism (i.e.,update trigger) to determine when an update record needs to be generated. Two type of subscriptions are introduced in [RFC8641], distinguished by how updates are triggered: periodic and on-change.

* Periodic subscription allows subscribed data to be streamed to the destination at a configured fixed periodic interval

* On-change subscription allows update to be triggered whenever a change in the subscribed information is detected. The periodic interval is set to zero value in the on-change subscription case.

However in some large scale deployments (e.g., wireless network performance monitoring) where an increased data collection rate is being used, it becomes more likely that a burst of streamed data may temporarily overwhelm a receiver and consume expensive network resource (e.g., radio resource). If the rate at which we can collect a stream of data is set too low or getting low priority telemetry data dropped, these telemetry data are not sufficient to detect and diagnose problems and verify correct network behavior.

There is a need for a service to configure both clients and servers with multiple different period intervals and corresponding subscription update policy which allows servers/publishers automatically switch to different period intervals according to resource usage change without the interaction with the client for policy update instruction, e.g., when the wireless signal strength falls below a configured low watermark, the subscribed data can be streamed at a higher rate while when the wireless signal strength crosses a configured high watermark, the subscribed data can be streamed at lower rate.

This document defines a YANG data model and associated mechanism enabling subscriber’s adaptive subscriptions to a publisher’s event streams. Applying these elements allows servers to automatically adjust the volume of telemetry traffic and rate of traffic sent from publisher to the receivers.

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 following terms are defined in [RFC5277] [RFC7950] [RFC3198] [RFC8342] [RFC8639] [RFC8641] and are not redefined here:

* Event

* Client

* Configuration

* Configured subscription

* Configuration datastore

* Notification message

* Publisher

* Receiver

* Subscriber

* Subscription

* On-change subscription

* Periodic subscription

* Selection filter

This document defines the following term:

Adaptive Subscription: Apply subscription update policy on the servers and allow servers/publishers automatically switch to different period intervals according to the resource usage change without the interaction with the client for update policy instruction.

2. Model Overview

This document defines a YANG module "ietf-adaptive-subscription", which augments the "update-trigger" choice defined in the "ietf-yang-push" module [RFC8641] with subscription configuration parameters that are specific to adaptive subscription.

In addition to Subscription state notifications defined in [RFC8639] and Notifications for Subscribed Content defined in [RFC8641], "ietf-adaptive-subscription" YANG module also defines "adaptive-period-update" notification to report update interval change.

The following tree diagrams [RFC8340] provide an overview of the data model for "ietf-adaptive-subscription.yang" module.

```
module: ietf-adaptive-subscription
  augment /sn:subscriptions/sn:subscription/yp:update-trigger:
    +--rw (adaptive-subscription)?
       +--:(adaptive-subscriptions)
          +--rw adaptive-subscriptions
              +--rw adaptive-period* [name]
                +--rw name                      string
                +--rw xpath-external-eval     string
                +--rw watermark?              uint32
                +--rw period                  centiseconds
                +--rw anchor-time?            yang:date-and-time
  augment /sn:establish-subscription/sn:input/yp:update-trigger:
    +-- (adaptive-subscription)?
       +--:(adaptive-subscriptions)
          +--rw adaptive-subscriptions
             +--rw adaptive-period* [name]
                +--rw name                    string
                +--rw xpath-external-eval     string
                +--rw watermark?              uint32
                +--rw period                  centiseconds
                +--rw anchor-time?            yang:date-and-time
  notifications:
    +---n adaptive-period-update
       +--ro id?                                  sn:subscription-id
       +--ro period                               centiseconds
       +--ro anchor-time?                         yang:date-and-time
       +--ro (selection-filter)?
          +--:(by-reference)
          |  +--ro selection-filter-ref           selection-filter-ref
          +--:(within-subscription)
             +--ro (filter-spec)?
                +--:(datastore-subtree-filter)
                |  +--ro datastore-subtree-filter?   <anydata> {sn:subtree}?
                +--:(datastore-xpath-filter)
                   +--ro datastore-xpath-filter?     yang:xpath1.0 {sn:xpath}?
```

2.1. Subscription Configuration

For adaptive subscriptions, triggered updates will occur at the boundaries of specified time intervals when a trigger condition is satisfied. These boundaries can be calculated from the adaptive periodic parameters:

* a "period" that defines the new duration between push updates, the period can be changed based on trigger condition.

* an "anchor-time" update intervals fall on the points in time that are a multiple of a "period" from an "anchor-time". If an "anchor-time" is not provided, then the "anchor-time" MUST be set with the creation time of the initial update record.

* a "watermark" that defines the threshold value of the targeted data object, e.g., it can be lower boundary or upper boundary of targeted data object.

* a "xpath-external-eval" represents a stanard XPath Evaluation criteria (See section 6.4 of [RFC7950]) that is applied against the targeted data object, which is used to trigger update interval switch in the server. It follows the rules defined in section 3.4 of [XPATH1.0] and contains comparisons of datastore node with its value to the specific threshold (i.e., watermark) and associated logical operation in the XPath format. Different from stream-xpath-filter defined in [RFC8639], it doesn't influence the event records output generation from a publisher.

2.2. YANG RPC

2.2.1. "establish-subscription" RPC

The augmentation of YANG module ietf-yang-push made to RPCs specified in YANG module ietf-subscribed-notifications [RFC8639] is introduced. This augmentation concerns the "establish- subscription" RPC, which is augmented with parameters that are needed to specify adaptive subscriptions. These parameters are same as one defined in Section 2.1.

2.3. Notifications for Adaptive Subscribed Content

The adaptive update notification is similar to Subscription state change notifications defined in [RFC8639]. It is inserted into the sequence of notification messages sent to a particular receiver. The adaptive update notification cannot be dropped or filtered out, it cannot be stored in replay buffers, and it is delivered only to impacted receivers of a subscription. The identification of adaptive update notification is easy to separate from other notification messages through the use of the YANG extension "subscription-state-notif". This extension tags a notification as a subscription state change notification.

The objects in the 'adpative-update' notification include:

* a "period" that defines the duration between push updates, the period can be changed based on trigger condition.

* an "anchor-time"; update intervals fall on the points in time that are a multiple of a "period" from an "anchor-time". If an "anchor-time" is not provided, then the "anchor-time" MUST be set with the creation time of the initial update record.

* A selection filter identifying YANG nodes of interest in a datastore. Filter contents are specified via a reference to an existing filter or via an in-line definition for only that subscription based on XPath Evaluation criteria defined in section 6.4 of [RFC7950]. Referenced filters allow an implementation avoid evaluating filter acceptability during a dynamic subscription request. The "case" statement differentiates the options. Note that filter contents are not affected by "xpath-external-eval" parameter and "watermark" parameter defined by update trigger.

3. Arbitrary XPath Complexity Evaluation

YANG-Push subscriptions [RFC8641] specifies selection filters to identify targeted YANG datastore nodes and/or datastore subtrees for which updates are to be pushed. In addition, it specifies update policies that contain conditions that trigger the generation and pushing of new update records. To support adaptive subscription defined in this document, the trigger condition can also use similar selection filter to express a standard XPath Evaluation criteria (section 6.4 of [RFC7950]) against targeted data object.

Similar to on change subscription, the adaptive subscription are particularly effective for data that changes infrequently, the following complex design choices need to be cautious, although these designs have already been well supported by the section 3.4 of [XPATH1.0]:

* Support XPath Evaluation criteria against every data objects;

* Support any type of node set in the XPath Evaluation criteria, e.g., string,int64, uint64, and decimal64 types;

* Both objects in the XPath Evaluation criteria to be compared are node-sets;

* Two objects to be compared are in different data type, e.g., one is integer, the other is string

As described in section 6.4 of RFC7950, Numbers in XPath 1.0 are IEEE 754 [IEEE754-2008] double-precision floating-point values; some values of int64, uint64, and decimal64 types cannot be exactly represented in XPath expressions.

If Two Objects to be compared are in different data type, conversion function is needed to convert different data type into numbers.

In both objects in the XPath Evaluation criteria to be compared are node-sets, more computation resources are required which add complexity.

To reduce these complexities, the following design principles are recommended:

* XPath Evaluation criteria against minimal set of data objects in the data model, these minimal set of data objects can be advertised using Notification capabilities model defined in [I-D.netconf-notification-capabilities].

* XPath Evaluation criteria only support condition expression that filter updates based on numbers.

* One object to be compared in the XPath Evaluation criteria MUST be nodeset.

* The other object to be compared in the XPath Evaluation criteria MUST be numbers data type.

4. Adaptive Subscription YANG Module

```
   <CODE BEGINS> file "ietf-adaptive-subscription@2020-02-14.yang"
   module ietf-adaptive-subscription {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription";
     prefix as;
     import ietf-subscribed-notifications {
       prefix sn;
     }
     import ietf-yang-push {
       prefix yp;
     }
     import ietf-yang-types {
       prefix yang;
     }

     organization
       "IETF NETCONF (Network Configuration) Working Group";
     contact
       "";
     description
       "NETCONF Protocol Data Types and Protocol Operations.
        Copyright (c) 2020 IETF Trust and the persons identified as
        the document authors.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

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

     revision 2019-12-15 {
       description
         "Initial revision";
       reference
         "RFCxxx Adaptive subscription to YANG notification.";
     }

     typedef centiseconds {
       type uint32;
       description
         "A period of time, measured in units of 0.01 seconds.";
     }

     typedef seconds {
       type uint32;
       description
         "A period of time, measured in units of 1 seconds.";
     }

     grouping adaptive-subscription-modifiable {
       description
         "This grouping describes the datastore-specific adaptive subscription
          conditions that can be changed during the lifetime of the
          subscription.";
       choice adaptive-subscription {
         description
           "Defines necessary conditions for sending an event record to
                  the subscriber.";
         container adaptive-subscriptions {
           list adaptive-period {
             key "name";
             description
               "Defines necessary conditions to switch update interval for
               sending an event record to the subscriber. The event record output
               generation will not be influeced these conditions.";
             leaf name {
               type string {
             length "1..64";
              }
           description
             "The name of the condition to be matched.  A device MAY further
              restrict the length of this name; space and special
              characters are not allowed.";
            }
             leaf xpath-external-eval {
               type string;
               description
                 "A XPath string, representing a logical expression,
                  which can contain comparisons of datastore values
                  and logical operations in the XPath format.";
             }
             leaf watermark {
               type uint32;
               description
                 "The watermark for targeted data object. The high
                  watermark, lowe watermark can be specified for the
                  targeted data object.";
             }
             leaf period {
               type centiseconds;
               mandatory true;
               description
                 "Duration of time that should occur between periodic
                  push updates, in units of 0.01 seconds.";
             }
             leaf anchor-time {
               type yang:date-and-time;
               description
                 "Designates a timestamp before or after which a series
                  of periodic push updates are determined.  The next
                  update will take place at a point in time that is a
                  multiple of a period from the 'anchor-time'.
                  For example, for an 'anchor-time' that is set for the
                  top of a particular minute and a period interval of a
                  minute, updates will be sent at the top of every
                  minute that this subscription is active.";
             }
           }
           description
             "Container for adaptive subscription.";
         }
       }
     }
```

```
     augment "/sn:subscriptions/sn:subscription/yp:update-trigger" {
       description
         "This augmentation adds additional subscription parameters
          that apply specifically to adaptive subscription.";
       uses adaptive-subscription-modifiable;
     }
     augment "/sn:establish-subscription/sn:input/yp:update-trigger" {
       description
         "This augmentation adds additional subscription parameters
            that apply specifically to datastore updates to RPC input.";
       uses adaptive-subscription-modifiable;
     }

     notification adaptive-period-update {
       sn:subscription-state-notification;
       description
         "This notification contains a push update that in turn contains
          data subscribed to via a subscription.  In the case of a
          periodic subscription, this notification is sent for periodic
          updates.  It can also be used for synchronization updates of
          an on-change subscription.  This notification shall only be
          sent to receivers of a subscription.  It does not constitute
          a general-purpose notification that would be subscribable as
          part of the NETCONF event stream by any receiver.";
       leaf id {
         type sn:subscription-id;
         description
           "This references the subscription that drove the
            notification to be sent.";
       }
       leaf period {
         type centiseconds;
         mandatory true;
         description
           "New duration of time that should occur between periodic
            push updates, in units of 0.01 seconds.";
       }
       leaf anchor-time {
         type yang:date-and-time;
         description
           "Designates a timestamp before or after which a series
            of periodic push updates are determined.  The next
            update will take place at a point in time that is a
            multiple of a period from the 'anchor-time'.
            For example, for an 'anchor-time' that is set for the
            top of a particular minute and a period interval of a
            minute, updates will be sent at the top of every
            minute that this subscription is active.";
       }
       uses yp:datastore-criteria {
         refine "selection-filter/within-subscription" {
           description
             "Specifies the selection filter and where it originated
              from.  If the 'selection-filter-ref' is populated, the
              filter in the subscription came from the 'filters'
              container.  Otherwise, it is populated in-line as part
              of the subscription itself.";
         }
       }
     }
   }
   <CODE ENDS>
```

5. IANA Considerations

5.1. Updates to the IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested to be made:

```
   ---------------------------------------------------------------------
      URI: urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription
      Registrant Contact: The IESG.
      XML: N/A, the requested URI is an XML namespace.
   ---------------------------------------------------------------------
```

5.2. Updates to the YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC7950]. . Following the format in [RFC6020], the following registration has been made:

```
   ---------------------------------------------------------------------
      Name:         ietf-adaptive-subscription
      Namespace:    urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription
      Prefix:       as
      Reference:    RFC xxxx
   ---------------------------------------------------------------------
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

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

* /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:watermark

* /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:period

* /sn:subscriptions/sn:subscription/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:anchor-time

* /sn:establish-subscription/sn:input/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:watermark

* /sn:establish-subscription/sn:input/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:period

* /sn:establish-subscription/sn:input/yp:update-trigger/as:adaptive-subscriptions/as:adaptive-period/as:anchor-time

7. Contributors

Michale Wang, Liang Geng for his major contributions to the initial modeling and use cases.

Michale Wang
Email: wangzitao@huawei.com

Liang Geng
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing 10053

Email: gengliang@chinamobile.com

8. Acknowledges

We would like to thanks Rob Wilton, Thomas Graf, Andy Bierman, Michael Richardson, Henk Birkholz for valuable review on this document, special thanks to Thmas and Michael to organize the discussion on several relevant drafts and reach the common understanding on the concept and ideas. Thanks Michael for providing CHIP/Matter WIFI statistics reference.

9. References

9.1. Normative References

[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>.

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

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

[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>.

[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>.

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

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

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

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

9.2. Informative References

[CHIP] CSA, "Connected Home over IP Specification", April 2021.

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

[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>.

[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>.

[XPATH1.0] W3C, "https://www.w3.org/TR/1999/REC-xpath-19991116/", 11 November 1999.

Appendix A. Example YANG Module

The example YANG module used in this document represents a Wi-Fi Network Diagnostics data specified in [CHIP] which can be used by a Node to assist a user or Administrative Node in diagnosing potential problems.

YANG tree diagram for the "example-wifi-network-diagnostic" module:

```
module: example-wifi-network-diagnostic
  +--rw server
  |  +--rw bssid?                       yang:mac-address
  |  +--rw security-type?               enumeration
  |  +--rw wifi-version?                enumeration
  |  +--rw channel-num?                 int8
  |  +--rw rssi?                        int8
  |  +--rw beacon-lost-count?           int8
  |  +--rw beacon-rx-count?             int8
  |  +--rw packet-multicast-rx-count?   int8
  |  +--rw packet-multicast-tx-count?   int8
  |  +--rw packet-unicast-rx-count?     int8
  |  +--rw packet-unicast-tx-count?     int8
  |  +--rw current-max-rate?            int8
  |  +--rw overrun-count?               int8
  +--rw events
     +--rw event* [name]
        +--rw name                   string
        +--rw disconnection?         enumeration
        +--rw association-failure?   enumeration
        +--rw connection-status?     enumeration
```

A.1. "example-wifi-mac" YANG Module

```
module example-wifi-network-diagnostic {
  yang-version 1;
  namespace "http://example.com/yang/wifi-network-diagnostic";
  prefix wnd;

  import ietf-yang-types {
    prefix yang;
  }

  container server {
    description
      "Configuration of the WiFi Server logical entity.";
```

```
       leaf bssid {
         type yang:mac-address;
         description
           "The MAC address of a wireless access point.";
       }
       leaf security-type {
         type enumeration {
           enum unspecified {
             value 0;
           }
           enum none {
             value 1;
           }
           enum wep {
             value 2;
           }
           enum wpa {
             value 3;
           }
           enum wpa2 {
             value 4;
           }
           enum wpa3 {
             value 5;
           }
         }
         description
           "The type of Wi-Fi security used. A value of 0
            indicate that the interface is not currently
            configured or operational.";
       }
       leaf wifi-version {
         type enumeration {
           enum 80211a {
             value 0;
           }
           enum 80211b {
             value 1;
           }
           enum 80211g {
             value 2;
           }
           enum 80211n {
             value 3;
           }
           enum 80211ac {
             value 4;
           }
           enum 80211ax {
             value 5;
           }
         }
         description
           "The highest 802.11 standard version usable
            by the Node.";
       }
       leaf channel-num {
         type int8;
         description
           "The channel that Wi-Fi communication is currently
            operating on. A value of 0indicates that the interface
            is not currently configured or operational.";
       }
       leaf rssi {
         type int8;
         description
           "The RSSI of the Nodes Wi-Fi radio in dBm.";
       }
       leaf beacon-lost-count {
         type int8;
         description
           "The count of the number of missed beacons the
            Node has detected.";
       }
       leaf beacon-rx-count {
         type int8;
         description
           "The count of the number of received beacons. The
            total number of expected beacons that could have been
            received during the interval since association SHOULD
            match the sum of BeaconRxCount and BeaconLostCount. ";
       }
       leaf packet-multicast-rx-count {
         type int8;
         description
           "The number of multicast packets received by
            the Node.";
       }
       leaf packet-multicast-tx-count {
         type int8;
         description
           "The number of multicast packets transmitted by
            the Node.";
       }
       leaf packet-unicast-rx-count {
         type int8;
         description
           "The number of multicast packets received by
            the Node.";
       }
       leaf packet-unicast-tx-count {
         type int8;
         description
           "The number of multicast packets transmitted by
            the Node.";
       }
       leaf current-max-rate {
         type int8;
         description
           "The current maximum PHY rate of transfer of
            data in bytes-per-second.";
       }
       leaf overrun-count {
         type int8;
         description
           "The number of packets dropped either at ingress or
            egress, due to lack of buffer memory to retain all
            packets on the ethernet network interface. The
            OverrunCount attribute SHALL be reset to 0 upon a
            reboot of the Node..";
       }
     }
     container events {
       description
         "Configuration of WIFI Network Diagnostic events.";
       list event {
         key "name";
         description
           "The list of event sources configured on the
            server.";
         leaf name {
           type string;
           description
             "The unique name of an event source.";
         }
         leaf disconnection {
           type enumeration {
             enum de-authenticated {
               value 1;
             }
             enum dis-association {
               value 2;
             }
           }
           description
             "A Nodes Wi-Fi connection has been disconnected as a
              result of de-authenticated or dis-association and
              indicates the reason.";
         }
         leaf association-failure {
           type enumeration {
             enum unknown {
               value 0;
             }
             enum association-failed {
               value 1;
             }
             enum authentication-failed {
               value 2;
             }
             enum ssid-not-found {
               value 3;
             }
           }
           description
             "A Node has attempted to connect, or reconnect, to
              a Wi-Fi access point, but is unable to successfully
              associate or authenticate, after exhausting all
              internal retries of its supplicant.";
         }
         leaf connection-status {
           type enumeration {
             enum connected {
               value 1;
             }
             enum notconnected {
               value 2;
             }
           }
           description
             "A Node’s connection status to a Wi-Fi network has
              changed. Connected, in this context, SHALL mean that
              a Node acting as a Wi-Fi station is successfully
              associated to a Wi-Fi Access Point.";
         }
       }
     }
   }
```

Appendix B. Adaptive Subscription and Notification Example

The examples within this document use the normative YANG module "ietf-adaptive-subscription" as defined in Section 4 and the non-normative example YANG module "example-wifi-network-diagnostic" as defined in Appendix A.1.

This section shows some typical adaptive subscription and notification message exchanges.

B.1. "edit-config" Example

The client configures adaptive subscription policy parameters on the server. The adaptive subscription configuration parameters require the server to support two update intervals (i.e., 5 seconds, 60 seconds) and scan all clients every 60 seconds in the sampling window if the rssi value of client is greater than or equal to -65dB in the sampling window; If the rssi value of client is less than -65dB, switch to 5 seconds period value, and then scan all clients every 60 seconds.

```
 <rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
   <target>
    <running/>
   </target>
   <config
    xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <top
     xmlns="http://example.com/schema/1.2/config"
     xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
     <yp:datastore
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
              ds:running
     </yp:datastore>
     <yp:datastore-xpath-filter
      xmlns:wnd="https://example.com/sample-data/1.0">
             /wnd:example-wifi-network-diagnostic
     </yp:datastore-xpath-filter>
     <as:adaptive-subscriptions
      xmlns:as="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
      <as:adaptive-period>
       <as:xpath-external-eval>
             /wnd:server[rssi &lt; -65]
      </as:xpath-external-eval>
      <as:watermark>-65</as:watermark>
       <as:period>5</as:period>
      </as:adaptive-period>
      <as:adaptive-period>
       <as:xpath-external-eval>
             /wnd:server[rssi &ge; -65]
          </as:xpath-external-eval>
       <as:watermark>-65</as:watermark>
       <as:period>60</as:period>
      </as:adaptive-period>
     </as:adaptive-subscriptions>
    </top>
   </config>
  </edit-config>
 </rpc>
```

B.2. Create Adaptive Subscription Example

The subscriber sends an "establish-subscription" RPC with the parameters listed in to request the creation of a adaptive subscription. The adaptive subscription configuration parameters require the server to scan all clients every 5 seconds if the rssi value of client is less than -65dB; If the rssi value of client is great than or equal to -65dB, switch to 60 seconds period value, and then report all clients every 60 seconds or scan every 5 seconds, collect 12 measurement values but report the last measurement value or average value of 12 measurement values. (Section 2)

```
<netconf:rpc message-id="101"
 xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
 <establish-subscription
  xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
  xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
  <yp:datastore
   xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
          ds:running
  </yp:datastore>
  <yp:datastore-xpath-filter
   xmlns:wnd="https://example.com/sample-data/1.0">
       /wnd:example-wifi-network-diagnostic
  </yp:datastore-xpath-filter>
  <as:adaptive-subscriptions
   xmlns:as="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
   <as:adaptive-period>
   <as:xpath-external-eval>wnd:server[rssi &lt; -65]
        </as:xpath-external-eval>
    <as:watermark>-65</as:watermark>
    <as:period>5</as:period>
   </as:adaptive-period>
   <as:adaptive-period>
    <as:xpath-external-eval>wnd:server[rssi &ge; -65]
    </as:xpath-external-eval>
    <as:watermark>-65</as:watermark>
    <as:period>60</as:period>
   </as:adaptive-period>
  </as:adaptive-subscriptions>
 </establish-subscription>
</netconf:rpc>
```

In another example, the adaptive subscription configuration parameters could also require the server to scan all clients every 5 seconds and report if the difference between maximum value of client rssi and minimum value of client rssi is greater than 0.20 dB in the sampling window; If the difference between maximum value of client rssi and minimum value of client rssi is less than 0.20 dB, switch to 60 seconds period value and then scan all clients every 60 seconds and report the last measurement value. If the difference between maximum value of client rssi and minimum value of client rssi is greater than or equal to 0.20 dB in two consecutive sampling windows, then in the second sampling window, only report the measurement value not reported by the previous sampling window.

```
<netconf:rpc message-id="101"
 xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
 <establish-subscription
  xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications"
  xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
  <yp:datastore
   xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
           ds:running
  </yp:datastore>
  <yp:datastore-xpath-filter
   xmlns:wnd="https://example.com/sample-data/1.0">
       /wnd:example-wifi-network-dianostic
  </yp:datastore-xpath-filter>
  <as:adaptive-subscriptions
   xmlns:as="urn:ietf:params:xml:ns:yang:ietf-adaptive-subscription">
   <as:adaptive-period>
    <as:xpath-external-eval>
        wnd:server[max(rssi)-min(rssi) &ge; 20]
         </as:xpath-external-eval>
    <as:watermark>20</as:watermark>
    <as:period>5</as:period>
   </as:adaptive-period>
   <as:adaptive-period>
    <as:xpath-external-eval>
        wnd:server[max(rssi)-min(rssi) &lt; 20]
    </as:xpath-external-eval>
    <as:watermark>20</as:watermark>
    <as:period>60</as:period>
   </as:adaptive-period>
  </as:adaptive-subscriptions>
 </establish-subscription>
</netconf:rpc>
```

B.3. "adaptive-period-update" notification example

Upon the server switches to from the update interval 5 seconds to the new update interval 60 seconds, Before sending event records to receivers, the "adaptive-update" notification should be generated and sent to the receivers to inform the receivers that the update interval value is switched to the new value.

```
   <notification
    xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0"
    xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <eventTime>2016-11-21T13:51:00Z</eventTime>
    <adaptive-period-update
     xmlns="http://example.com/ietf-adaptive-subscription">
     <id>0</id>
     <period>60</period>
     <yp:datastore
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
          ds:running
     </yp:datastore>
     <yp:datastore-xpath-filter
      xmlns:ex="https://example.com/sample-data/1.0">
          /ex:example-wifi-network-diagnostic
     </yp:datastore-xpath-filter>
    </adaptive-period-update>
   </notification>
```

B.4. Changes between Revisions

v06 -v07

* The usage examples typo fixed in the Appendix.

* Add reference to RFC7950 XPATH Evaluation section and XPATH 1.0

* Clarify the definitions of 'xpath-external-eval' and 'selection-filter' by reusing XPATH Evaluation rules in RFC7950.

* Add a new terminology "adaptive subscription".

* Add one section to discuss Arbitrary XPath Complexity.

v05 -v06

* Replace example-wifi-mac module with example-wifi-network-diagnostic using WIFI statistics specified in CHIP specification.

* Update adaptive subscription Example to align with WIFI example module change.

* Add one more reference to CHIP Specification.

v04 -v05

* Remove "modify-subscption" RPC usage.

* Module update to fix the nits.

* Update adaptive subscription Example.

* Other Editorial changes.

v03 - v04

* Add missing subtrees and data nodes in the security section;

* Change "adaptive-update" notification into "adaptive-period-update" notification;

* Other Editorial changes.

v02 - v03

* Clarify the difference between low priority telemetry data dropping and collection rate switching in the introduction section;

* Update the abstract and introduction section to focus on collection rate switching in the server without interaction with the remote client;

* Format usage example and change ssid into rssi in the appendix;

* Use boilerplate and reuse the terms in the terminology section.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: bill.wu@huawei.com

Wei Song
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: songwei80@huawei.com

Peng Liu
China Mobile
32 Xuanwumen West St, Xicheng District
Beijing

Email: liupengyjy@chinamobile.com

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: maqiufang1@huawei.com

Wei Wang
China Telecom
32 Xuanwumen West St, Xicheng District
Beijing

Email: wangw36@chinatelecom.cn

NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 April 2022

K. Watsen
Watsen Networks
Q. Wu
Huawei Technologies
O. Hagsand
Netgate
H. Li
Hewlett Packard Enterprise
P. Andersson
Cisco Systems

25 October 2021

List Pagination for YANG-driven Protocols
draft-wwlh-netconf-list-pagination-00

Abstract

In some circumstances, instances of YANG modeled "list" and "leaf-list" nodes may contain numerous entries. Retrieval of all the entries can lead to inefficiencies in the server, the client, and the network in between.

This document defines a model for list pagination that can be implemented by YANG-driven management protocols such as NETCONF and RESTCONF. The model supports paging over optionally filtered and/or sorted entries. The solution additionally enables servers to constrain query expressions on some "config false" lists or leaf-lists.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

YANG modeled "list" and "leaf-list" nodes may contain a large number of entries. For instance, there may be thousands of entries in the configuration for network interfaces or access control lists. And time-driven logging mechanisms, such as an audit log or a traffic log, can contain millions of entries.

Retrieval of all the entries can lead to inefficiencies in the server, the client, and the network in between. For instance, consider the following:

* A client may need to filter and/or sort list entries in order to, e.g., present the view requested by a user.

* A server may need to iterate over many more list entries than needed by a client.

* A network may need to convey more data than needed by a client.

Optimal global resource utilization is obtained when clients are able to cherry-pick just that which is needed to support the application-level business logic.

This document defines a generic model for list pagination that can be implemented by YANG-driven management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Details for how such protocols are updated are outside the scope of this document.

The model presented in this document supports paging over optionally filtered and/or sorted entries. Server-side filtering and sorting is ideal as servers can leverage indexes maintained by a backend storage layer to accelerate queries.

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 following terms are defined in [RFC7950] and are not redefined here: client, data model, data tree, feature, extension, module, leaf, leaf-list, and server.

1.2. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

1.3. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. The "ietf-list-pagination" module only defines a YANG extension and augments a couple leafs into a "config false" node defined by the "ietf-system-capabilities" module.

2. Solution Overview

The solution presented in this document broadly entails a client sending a query to a server targeting a specific list or leaf-list including optional parameters guiding which entries should be returned.

A secondary aspect of this solution entails a client sending a query parameter to a server guiding how descendent lists and leaf-lists should be returned. This parameter may be used on any target node, not just "list" and "leaf-list" nodes.

Clients detect a server's support for list pagination via an entry for the "ietf-list-pagination" module (defined in Section 4) in the server's YANG Library [RFC8525] response.

Relying on client-provided query parameters ensures servers remain backward compatible with legacy clients.

3. Solution Details

This section is composed of the following subsections:

* Section 3.1 defines five query parameters clients may use to page through the entries of a single list or leaf-list in a data tree.

* Section 3.2 defines one query parameter that clients may use to affect the content returned for descendant lists and leaf-lists.

* Section 3.3 defines per schema-node tags enabling servers to indicate which "config false" lists are constrained and how they may be interacted with.

3.1. Query Parameters for a Targeted List or Leaf-List

The five query parameters presented this section are listed in processing order. This processing order is logical, efficient, and matches the processing order implemented by database systems, such as SQL.

The order is as follows: a server first processes the "where" parameter (see Section 3.1.1), then the "sort-by" parameter (see Section 3.1.2), then the "direction" parameter (see Section 3.1.3), then the "offset" parameter (see Section 3.1.4), and lastly the "limit" parameter (see Section 3.1.5).

3.1.1. The "where" Query Parameter

Description
The "where" query parameter specifies a filter expression that result-set entries must match.

Default Value
If this query parameter is unspecified, then no entries are filtered from the working result-set.

Allowed Values
The allowed values are XPath 1.0 expressions. It is an error if the XPath expression references a node identifier that does not exist in the schema, is optional or conditional in the schema or, for constrained "config false" lists and leaf-lists (see Section 3.3), if the node identifier does not point to a node having the "indexed" extension statement applied to it (see Section 3.3.2).

Conformance
The "where" query parameter MUST be supported for all "config true" lists and leaf-lists and SHOULD be supported for "config false" lists and leaf-lists. Servers MAY disable the support for some or all "config false" lists and leaf-lists as described in Section 3.3.2.

3.1.2. The "sort-by" Query Parameter

Description
The "sort-by" query parameter indicates the node in the working result-set (i.e., after the "where" parameter has been applied) that entries should be sorted by. Sorts are in ascending order (e.g., '1' before '9', 'a' before 'z', etc.). Missing values are sorted to the end (e.g., after all nodes having values). Sub-sorts are not supported.

Default Value
If this query parameter is unspecified, then the list or leaf-list's default order is used, per the YANG "ordered-by" statement (see Section 7.7.7 of [RFC7950]).

Allowed Values
The allowed values are node identifiers. It is an error if the specified node identifier does not exist in the schema, is optional or conditional in the schema or, for constrained "config false" lists and leaf-lists (see Section 3.3), if the node identifier does not point to a node having the "indexed" extension statement applied to it (see Section 3.3.2).

Conformance
The "sort-by" query parameter MUST be supported for all "config true" lists and leaf-lists and SHOULD be supported for "config false" lists and leaf-lists. Servers MAY disable the support for some or all "config false" lists and leaf-lists as described in Section 3.3.2.

3.1.3. The "direction" Query Parameter

Description
The "direction" query parameter indicates how the entries in the working result-set (i.e., after the "sort-by" parameter has been applied) should be traversed.

Default Value
If this query parameter is unspecified, the default value is "forwards".

Allowed Values
The allowed values are:

forwards
Return entries in the forwards direction. Also known as the "default" or "ascending" direction.

backwards
Return entries in the backwards direction. Also known as the "reverse" or "descending" direction

Conformance
The "direction" query parameter MUST be supported for all lists and leaf-lists.

3.1.4. The "offset" Query Parameter

Description
The "offset" query parameter indicates the number of entries in the working result-set (i.e., after the "direction" parameter has been applied) that should be skipped over when preparing the response.

Default Value
If this query parameter is unspecified, then no entries in the result-set are skipped, same as when the offset value '0' is specified.

Allowed Values
The allowed values are unsigned integers. It is an error for the offset value to exceed the number of entries in the working result-set, and the "offset-out-of-range" identity SHOULD be produced in the error output when this occurs.

Conformance
The "offset" query parameter MUST be supported for all lists and leaf-lists.

3.1.5. The "limit" Query Parameter

Description
The "limit" query parameter limits the number of entries returned from the working result-set (i.e., after the "offset" parameter has been applied). Any list or leaf-list that is limited includes, somewhere in its encoding, a metadata value [RFC7952] called "remaining", a positive integer indicating the number of elements that were not included in the result-set by the "limit" operation, or the value "unknown" in case, e.g., the server determines that counting would be prohibitively expensive.

Default Value
If this query parameter is unspecified, the number of entries that may be returned is unbounded.

Allowed Values
The allowed values are positive integers.

Conformance
The "limit" query parameter MUST be supported for all lists and leaf-lists.

3.2. Query Parameter for Descendant Lists and Leaf-Lists

Whilst this document primarily regards pagination for a list or leaf-list, it begs the question for how descendant lists and leaf-lists should be handled, which is addressed by the "sublist-limit" query parameter described in this section.

3.2.1. The "sublist-limit" Query Parameter

Description
The "sublist-limit" parameter limits the number of entries returned for descendent lists and leaf-lists.

Any descendent list or leaf-list limited by the "sublist-limit" parameter includes, somewhere in its encoding, a metadata value [RFC7952] called "remaining", a positive integer indicating the number of elements that were not included by the "sublist-limit" parameter, or the value "unknown" in case, e.g., the server determines that counting would be prohibitively expensive.

When used on a list node, it only affects the list’s descendant nodes, not the list itself, which is only affected by the parameters presented in Section 3.1.

Default Value
   If this query parameter is unspecified, the number of entries that may be returned for descendent lists and leaf-lists is unbounded.

Allowed Values
   The allowed values are positive integers.

Conformance
   The "sublist-limit" query parameter MUST be supported for all conventional nodes, including a datastore's top-level node (i.e., '/').

3.3. Constraints on "where" and "sort-by" for "config false" Lists

Some "config false" lists and leaf-lists may contain an enormous number of entries. For instance, a time-driven logging mechanism, such as an audit log or a traffic log, can contain millions of entries.

In such cases, "where" and "sort-by" expressions will not perform well if the server must bring each entry into memory in order to process it.

The server's best option is to leverage query-optimizing features (e.g., indexes) built into the backend database holding the dataset.

However, arbitrary "where" expressions and "sort-by" node identifiers into syntax supported by the backend database and/or query-optimizers may prove challenging, if not impossible, to implement.

Thusly this section introduces mechanisms whereby a server can:

1. Identify which "config false" lists and leaf-lists are constrained.

2. Identify what node-identifiers and expressions are allowed for the constrained lists and leaf-lists.

> Note: The pagination performance for "config true" lists and leaf-lists is not considered as already servers must be able to process them as configuration. Whilst some "config true' lists and leaf-lists may contain thousands of entries, they are well within the capability of server-side processing.

3.3.1. Identifying Constrained "config false" Lists and Leaf-Lists

Identification of which lists and leaf-lists are constrained occurs in the schema tree, not the data tree. However, as server abilities vary, it is not possible to define constraints in YANG modules defining generic data models.

In order to enable servers to identify which lists and leaf-lists are constrained, the solution presented in this document augments the data model defined by the "ietf-system-capabilities" module presented in [I-D.ietf-netconf-notification-capabilities].

Specifically, the "ietf-list-pagination" module (see Section 4) augments an empty leaf node called "constrained" into the "per-node-capabilities" node defined in the "ietf-system-capabilities" module.

The "constrained" leaf MAY be specified for any "config false" list or leaf-list.

When a list or leaf-list is constrained:

* All parts of XPath 1.0 expressions are disabled unless explicitly enabled by Section 3.3.2.

* Node-identifiers used in "where" expressions and "sort-by" filters MUST have the "indexed" leaf applied to it (see Section 3.3.2).

* For lists only, node-identifiers used in "where" expressions and "sort-by" filters MUST NOT descend past any descendent lists. This ensures that only indexes relative to the targeted list are used. Further constraints on node identifiers MAY be applied in Section 3.3.2.

3.3.2. Indicating the Constraints for "where" Filters and "sort-by" Expressions

This section identifies how constraints for "where" filters and "sort-by" expressions are specified. These constraints are valid only if the "constrained" leaf described in the previous section Section 3.3.1 has been set on the immediate ancestor "list" node or, for "leaf-list" nodes, on itself.

3.3.2.1. Indicating Filterable/Sortable Nodes

For "where" filters, an unconstrained XPath expressions may use any node in comparisons. However, efficient mappings to backend databases may support only a subset of the nodes.

Similarly, for "sort-by" expressions, efficient sorts may only support a subset of the nodes.

In order to enable servers to identify which nodes may be used in comparisons (for both "where" and "sort-by" expressions), the "ietf-list-pagination" module (see Section 4) augments an empty leaf node called "indexed" into the "per-node-capabilities" node defined in the "ietf-system-capabilities" module (see [I-D.ietf-netconf-notification-capabilities]).

When a "list" or "leaf-list" node has the "constrained" leaf, only nodes having the "indexed" node may be used in "where" and/or "sort-by" expressions. If no nodes have the "indexed" leaf, when the "constrained" leaf is present, then "where" and "sort-by" expressions are disabled for that list or leaf-list.

4. The "ietf-list-pagination" Module

The "ietf-list-pagination" module is used by servers to indicate that they support pagination on YANG "list" and "leaf-list" nodes, and to provide an ability to indicate which "config false" list and/or "leaf-list" nodes are constrained and, if so, which nodes may be used in "where" and "sort-by" expressions.

4.1. Data Model Overview

The following tree diagram [RFC8340] illustrates the "ietf-list-pagination" module:

```
module: ietf-list-pagination

  augment /sysc:system-capabilities/sysc:datastore-capabilities
            /sysc:per-node-capabilities:
    +--ro constrained?   empty
    +--ro indexed?       empty
```

Comments:

* As shown, this module augments two optional leaves into the "node-selector" node of the "ietf-system-capabilities" module.

* Not shown is that the module also defines an "md:annotation" statement named "remaining". This annotation may be present in a server's response to a client request containing either the "limit" (Section 3.1.5) or "sublist-limit" parameters (Appendix A.3.6).

4.2. Example Usage

4.2.1. Constraining a "config false" list

The following example illustrates the "ietf-list-pagination" module's augmentations of the "system-capabilities" data tree. This example assumes the "example-social" module defined in the Appendix A.1 is implemented.

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

<system-capabilities
  xmlns="urn:ietf:params:xml:ns:yang:ietf-system-capabilities"
  xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores"
  xmlns:es="http://example.com/ns/example-social"
  xmlns:lpg="urn:ietf:params:xml:ns:yang:ietf-list-pagination">
  <datastore-capabilities>
    <datastore>ds:operational</datastore>
    <per-node-capabilities>
      <node-selector>/es:audit-logs/es:audit-log</node-selector>
      <lpg:constrained/>
    </per-node-capabilities>
    <per-node-capabilities>
      <node-selector>/es:audit-logs/es:audit-log/es:timestamp</node-\
selector>
      <lpg:indexed/>
    </per-node-capabilities>
    <per-node-capabilities>
      <node-selector>/es:audit-logs/es:audit-log/es:member-id</node-\
selector>
      <lpg:indexed/>
    </per-node-capabilities>
    <per-node-capabilities>
      <node-selector>/es:audit-logs/es:audit-log/es:outcome</node-se\
lector>
      <lpg:indexed/>
    </per-node-capabilities>
  </datastore-capabilities>
</system-capabilities>
```

4.2.2. Indicating number remaining in a limited list

FIXME: valid syntax for 'where'?

4.3. YANG Module

This YANG module has normative references to [RFC7952] and [I-D.ietf-netconf-notification-capabilities].

```
   <CODE BEGINS> file "ietf-list-pagination@2021-10-25.yang"

   module ietf-list-pagination {
     yang-version 1.1;
     namespace
       "urn:ietf:params:xml:ns:yang:ietf-list-pagination";
     prefix lpg;

     import ietf-yang-types {
       prefix yang;
       reference
         "RFC 6991: Common YANG Data Types";
     }

     import ietf-yang-metadata {
       prefix md;
       reference
         "RFC 7952: Defining and Using Metadata with YANG";
     }

     import ietf-system-capabilities  {
       prefix sysc;
       reference
         "draft-ietf-netconf-notification-capabilities:
          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>";

     description
       "This module is used by servers to 1) indicate they support
        pagination on 'list' and 'leaf-list' resources, 2) define a
        grouping for each list-pagination parameter, and 3) indicate
        which 'config false' lists have constrained 'where' and
        'sort-by' parameters and how they may be used, if at all.

        Copyright (c) 2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Simplified
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX
        (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

     revision 2021-10-25 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: List Pagination for YANG-driven Protocols";
     }


     // Annotations

     md:annotation remaining {
       type union {
         type uint32;
         type enumeration {
           enum "unknown" {
             description
               "Indicates that number of remaining entries is unknown
                to the server in case, e.g., the server has determined
                that counting would be prohibitively expensive.";
           }
         }
       }
       description
         "This annotation contains the number of elements not included
          in the result set (a positive value) due to a 'limit' or
          'sublist-limit' operation.  If no elements were removed,
          this annotation MUST NOT appear.  The minimum value (0),
          which never occurs in normal operation, is reserved to
          represent 'unknown'.  The maximum value (2^32-1) is
          reserved to represent any value greater than or equal
          to 2^32-1 elements.";
     }
     // Identities

     identity list-pagination-error {
       description
         "Base identity for list-pagination errors.";
     }

     identity offset-out-of-range {
       base list-pagination-error;
       description
         "The 'offset' query parameter value is greater than the number
          of instances in the target list or leaf-list resource.";
     }

     // Groupings

     grouping where-param-grouping {
       description
         "This grouping may be used by protocol-specific YANG modules
          to define a protocol-specific query parameter.";
       leaf where {
         type union {
           type yang:xpath1.0;
           type enumeration {
             enum "unfiltered" {
               description
                 "Indicates that no entries are to be filtered
                  from the working result-set.";
             }
           }
         }
         default "unfiltered";
         description
           "The 'where' parameter specifies a boolean expression
            that result-set entries must match.

            It is an error if the XPath expression references a node
            identifier that does not exist in the schema, is optional
            or conditional in the schema or, for constrained 'config
            false' lists and leaf-lists, if the node identifier does
            not point to a node having the 'indexed' extension
            statement applied to it (see RFC XXXX).";
       }
     }

     grouping sort-by-param-grouping {
       description
         "This grouping may be used by protocol-specific YANG modules
```

```
          to define a protocol-specific query parameter.";
       leaf sort-by {
         type union {
           type string {
             // An RFC 7950 'descendant-schema-nodeid'.
             pattern '([0-9a-fA-F]*:)?[0-9a-fA-F]*'
                     + '(/([0-9a-fA-F]*:)?[0-9a-fA-F]*)*';
           }
           type enumeration {
             enum "none" {
               description
                 "Indicates that the list or leaf-list's default
                  order is to be used, per the YANG 'ordered-by'
                  statement.";
             }
           }
         }
         default "none";
         description
           "The 'sort-by' parameter indicates the node in the
            working result-set (i.e., after the 'where' parameter
            has been applied) that entries should be sorted by.

            Sorts are in ascending order (e.g., '1' before '9',
            'a' before 'z', etc.).  Missing values are sorted to
            the end (e.g., after all nodes having values).";
       }
     }

     grouping direction-param-grouping {
       description
         "This grouping may be used by protocol-specific YANG modules
          to define a protocol-specific query parameter.";
       leaf direction {
         type enumeration {
           enum forwards {
             description
                "Indicates that entries should be traversed from
                 the first to last item in the working result set.";
           }
           enum backwards {
             description
                "Indicates that entries should be traversed from
                 the last to first item in the working result set.";
           }
         }
         default "forwards";
         description
```

```
           "The 'direction' parameter indicates how the entries in the
            working result-set (i.e., after the 'sort-by' parameter
            has been applied) should be traversed.";
       }
     }

     grouping offset-param-grouping {
       description
         "This grouping may be used by protocol-specific YANG modules
          to define a protocol-specific query parameter.";
       leaf offset {
         type uint32;
         default 0;
         description
           "The 'offset' parameter indicates the number of entries
            in the working result-set (i.e., after the 'direction'
            parameter has been applied) that should be skipped over
            when preparing the response.";
       }
     }

     grouping limit-param-grouping {
       description
         "This grouping may be used by protocol-specific YANG modules
          to define a protocol-specific query parameter.";
       leaf limit {
         type union {
           type uint32 {
             range "1..max";
           }
           type enumeration {
             enum "unbounded" {
               description
                 "Indicates that the number of entries that may be
                  returned is unbounded.";
             }
           }
         }
         default "unbounded";
         description
           "The 'limit' parameter limits the number of entries returned
            from the working result-set (i.e., after the 'offset'
            parameter has been applied).

            Any result-set that is limited includes, somewhere in its
            encoding, the metadata value 'remaining' to indicate the
            number entries not included in the result set.";
       }
     }

     grouping sublist-limit-param-grouping {
       description
         "This grouping may be used by protocol-specific YANG modules
          to define a protocol-specific query parameter.";
       leaf sublist-limit {
         type union {
           type uint32 {
             range "1..max";
           }
           type enumeration {
             enum "unbounded" {
               description
                 "Indicates that the number of entries that may be
                  returned is unbounded.";
             }
           }
         }
         default "unbounded";
         description
           "The 'sublist-limit' parameter limits the number of entries
            for descendent lists and leaf-lists.

            Any result-set that is limited includes, somewhere in
            its encoding, the metadata value 'remaining' to indicate
            the number entries not included in the result set.";
       }
     }

     // Protocol-accessible nodes

     augment // FIXME: ensure datastore == <operational>
       "/sysc:system-capabilities/sysc:datastore-capabilities"
       + "/sysc:per-node-capabilities" {
       description
         "Defines some leafs that MAY be used by the server to
          describe constraints imposed of the 'where' filters and
          'sort-by' parameters used in list pagination queries.";
       leaf constrained {
         type empty;
         description
           "Indicates that 'where' filters and 'sort-by' parameters
            on the targeted 'config false' list node are constrained.
            If a list is not 'constrained', then full XPath 1.0
            expressions may be used in 'where' filters and all node
            identifiers are usable by 'sort-by'.";
       }
       leaf indexed {
         type empty;
         description
           "Indicates that the targeted  descendent node of a
            'constrained' list (see the 'constrained' leaf) may be
            used in 'where' filters and/or 'sort-by' parameters.
            If a descendent node of a 'constrained' list is not
            'indexed', then it MUST NOT be used in 'where' filters
            or 'sort-by' parameters.";
       }
     }
   }

   <CODE ENDS>
```

5. IANA Considerations

5.1. The "IETF XML" Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688] maintained at https://www.iana.org/assignments/xml-registry/xml-registry.xhtml#ns. Following the format in [RFC3688], the following registration is requested:

```
   URI: urn:ietf:params:xml:ns:yang:ietf-list-pagination
   Registrant Contact: The IESG.
   XML: N/A, the requested URI is an XML namespace.
```

5.2. The "YANG Module Names" Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020] maintained at https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml. Following the format defined in [RFC6020], the below registration is requested:

```
   name: ietf-list-pagination
   namespace: urn:ietf:params:xml:ns:yang:ietf-list-pagination
   prefix: lpg
   RFC: XXXX
```

6. Security Considerations

6.1. Regarding the "ietf-list-pagination" YANG Module

Pursuant the template defined in ...FIXME

7. References

7.1. Normative References

[I-D.ietf-netconf-notification-capabilities]
Lengyel, B., Clemm, A., and B. Claise, "YANG Modules describing Capabilities for Systems and Datastore Update Notifications", Work in Progress, Internet-Draft, draft-ietf-netconf-notification-capabilities-21, 15 October 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-notification-capabilities-21>.

[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>.

[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>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[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>.

7.2. Informative References

[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>.

[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>.

[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>.

[RFC8525] Bierman, A., Bjorklund, M., Schoenwaelder, J., Watsen, K., and R. Wilton, "YANG Library", RFC 8525, DOI 10.17487/RFC8525, March 2019, <https://www.rfc-editor.org/info/rfc8525>.

Appendix A. Vector Tests

This normative appendix section illustrates every notable edge condition conceived during this document’s production.

Test inputs and outputs are provided in a manner that is both generic and concise.

Management protocol specific documents need only reproduce as many of these tests as necessary to convey pecularities presented by the protocol.

Implementations are RECOMMENDED to implement the tests presented in this document, in addition to any tests that may be presented in protocol specific documents.

A.1. Example YANG Module

The vector tests assume the "example-social" YANG module defined in this section.

This module has been specially crafted to cover every notable edge condition, especially with regards to the types of the data nodes.

Following is the tree diagram [RFC8340] for the "example-social" module:

```
   module: example-social
     +--rw members
     |  +--rw member* [member-id]
     |     +--rw member-id           string
     |     +--rw email-address       inet:email-address
     |     +--rw password            ianach:crypt-hash
     |     +--rw avatar?             binary
     |     +--rw tagline?            string
     |     +--rw privacy-settings
     |     |  +--rw hide-network?      boolean
     |     |  +--rw post-visibility?   enumeration
     |     +--rw following*          -> /members/member/member-id
     |     +--rw posts
     |     |  +--rw post* [timestamp]
     |     |     +--rw timestamp    yang:date-and-time
     |     |     +--rw title?       string
     |     |     +--rw body         string
     |     +--rw favorites
     |     |  +--rw uint8-numbers*       uint8
     |     |  +--rw uint64-numbers*      uint64
     |     |  +--rw int8-numbers*        int8
     |     |  +--rw int64-numbers*       int64
     |     |  +--rw decimal64-numbers*   decimal64
     |     |  +--rw bits*                bits
     |     +--ro stats
     |        +--ro joined              yang:date-and-time
     |        +--ro membership-level    enumeration
     |        +--ro last-activity?      yang:date-and-time
     +--ro audit-logs
        +--ro audit-log* []
           +--ro timestamp    yang:date-and-time
           +--ro member-id    string
           +--ro source-ip    inet:ip-address
           +--ro request      string
           +--ro outcome      boolean
```

Following is the YANG [RFC7950] for the "example-social" module:

```
   module example-social {
     yang-version 1.1;
     namespace "http://example.com/ns/example-social";
     prefix es;

     import ietf-yang-types {
       prefix yang;
       reference
         "RFC 6991: Common YANG Data Types";
     }
     import ietf-inet-types {
       prefix inet;
       reference
         "RFC 6991: Common YANG Data Types";
     }

     import iana-crypt-hash {
       prefix ianach;
       reference
         "RFC 7317: A YANG Data Model for System Management";
     }

     organization "Example, Inc.";
     contact      "support@example.com";
     description  "Example Social Data Model.";

     revision YYYY-MM-DD {
       description
         "Initial version.";
       reference
         "RFC XXXX: Example social module.";
     }

     container members {
       description
         "Container for list of members.";
       list member {
         key "member-id";
         description
           "List of members.";

         leaf member-id {
           type string {
             length "1..80";
             pattern '.*[\n].*' {
              modifier invert-match;
             }
           }
           description
             "The member's identifier.";
         }

         leaf email-address {
           type inet:email-address;
           mandatory true;
           description
             "The member's email address.";
         }
         leaf password {
           type ianach:crypt-hash;
           mandatory true;
           description
             "The member’s hashed-password.";
         }

         leaf avatar {
           type binary;
           description
             "An binary image file.";
         }

         leaf tagline {
           type string {
             length "1..80";
             pattern ’.*[\n].*’ {
               modifier invert-match;
             }
           }
           description
             "The member’s tagline.";
         }

         container privacy-settings {
           leaf hide-network {
             type boolean;
             description
               "Hide who you follow and who follows you.";
           }
           leaf post-visibility {
             type enumeration {
               enum public {
                 description
                   "Posts are public.";
               }
               enum unlisted {
                 description
                   "Posts are unlisted, though visable to all.";
               }
               enum followers-only {
                 description
                   "Posts only visible to followers.";
               }
             }
             default public;
             description
               "The post privacy setting.";
           }
           description
             "Preferences for the member.";
         }

         leaf-list following {
           type leafref {
             path "/members/member/member-id";
           }
           description
             "Other members this members is following.";
         }

         container posts {
           description
             "The member’s posts.";
           list post {
             key timestamp;
             leaf timestamp {
               type yang:date-and-time;
               description
                 "The timestamp for the member’s post.";
             }
             leaf title {
               type string {
                 length "1..80";
                 pattern ’.*[\n].*’ {
                   modifier invert-match;
                 }
               }
               description
                 "A one-line title.";
             }
             leaf body {
               type string;
               mandatory true;
               description
                 "The body of the post.";
             }
             description
               "A list of posts.";
           }
         }

         container favorites {
           description
             "The member’s favorites.";
           leaf-list uint8-numbers {
             type uint8;
             ordered-by user;
             description
               "The member's favorite uint8 numbers.";
           }
           leaf-list uint64-numbers {
             type uint64;
             ordered-by user;
             description
               "The member's favorite uint64 numbers.";
           }
           leaf-list int8-numbers {
             type int8;
             ordered-by user;
             description
               "The member's favorite int8 numbers.";
           }
           leaf-list int64-numbers {
             type int64;
             ordered-by user;
             description
               "The member's favorite uint64 numbers.";
           }
           leaf-list decimal64-numbers {
             type decimal64 {
               fraction-digits 5;
             }
             ordered-by user;
             description
               "The member's favorite decimal64 numbers.";
           }
           leaf-list bits {
             type bits {
               bit zero {
                 position 0;
                 description "zero";
               }
               bit one {
                 position 1;
                 description "one";
               }
               bit two {
                 position 2;
                 description "two";
               }
             }
             ordered-by user;
             description
```

```
                "The member's favorite bits.";
            }
          }

          container stats {
            config false;
            description
              "Operational state members values.";
            leaf joined {
              type yang:date-and-time;
              mandatory true;
              description
                "Timestamp when member joined.";
            }
            leaf membership-level {
              type enumeration {
                enum admin {
                  description
                    "Site administrator.";
                }
                enum standard {
                  description
                    "Standard membership level.";
                }
                enum pro {
                  description
                    "Professional membership level.";
                }
              }
              mandatory true;
              description
                "The membership level for this member.";
            }
            leaf last-activity {
              type yang:date-and-time;
              description
                "Timestamp of member's last activity.";
            }
          }
        }
      }

      container audit-logs {
        config false;
        description
          "Audit log configuration";
        list audit-log {
          description
```

```
           "List of audit logs.";
         leaf timestamp {
           type yang:date-and-time;
           mandatory true;
           description
             "The timestamp for the event.";
         }
         leaf member-id {
           type string;
           mandatory true;
           description
             "The 'member-id' of the member.";
         }
         leaf source-ip {
           type inet:ip-address;
           mandatory true;
           description
             "The apparent IP address the member used.";
         }
         leaf request {
           type string;
           mandatory true;
           description
             "The member's request.";
         }
         leaf outcome {
           type boolean;
           mandatory true;
           description
             "Indicate if request was permitted.";
         }
       }
     }
   }
```

A.2. Example Data Set

The examples assume the server's operational state as follows.

The data is provided in JSON only for convenience and, in particular, has no bearing on the "generic" nature of the tests themselves.

```
   {
     "example-social:members": {
       "member": [
         {
           "member-id": "bob",
           "email-address": "bob@example.com",
            "password": "$0$1543",
            "avatar": "BASE64VALUE=",
            "tagline": "Here and now, like never before.",
            "posts": {
              "post": [
                {
                  "timestamp": "2020-08-14T03:32:25Z",
                  "body": "Just got in."
                },
                {
                  "timestamp": "2020-08-14T03:33:55Z",
                  "body": "What’s new?"
                },
                {
                  "timestamp": "2020-08-14T03:34:30Z",
                  "body": "I’m bored..."
                }
              ]
            },
            "favorites": {
              "decimal64-numbers": ["3.14159", "2.71828"]
            },
            "stats": {
              "joined": "2020-08-14T03:30:00Z",
              "membership-level": "standard",
              "last-activity": "2020-08-14T03:34:30Z"
            }
          },
          {
            "member-id": "eric",
            "email-address": "eric@example.com",
            "password": "$0$1543",
            "avatar": "BASE64VALUE=",
            "tagline": "Go to bed with dreams; wake up with a purpose.",
            "following": ["alice"],
            "posts": {
              "post": [
                {
                  "timestamp": "2020-09-17T18:02:04Z",
                  "title": "Son, brother, husband, father",
                  "body": "What’s your story?"
                }
              ]
            },
            "favorites": {
              "bits": ["two", "one", "zero"]
            },
            "stats": {
             "joined": "2020-09-17T19:38:32Z",
             "membership-level": "pro",
             "last-activity": "2020-09-17T18:02:04Z"
           }
         },
         {
           "member-id": "alice",
           "email-address": "alice@example.com",
           "password": "$0$1543",
           "avatar": "BASE64VALUE=",
           "tagline": "Every day is a new day",
           "privacy-settings": {
             "hide-network": "false",
             "post-visibility": "public"
           },
           "following": ["bob", "eric", "lin"],
           "posts": {
             "post": [
               {
                 "timestamp": "2020-07-08T13:12:45Z",
                 "title": "My first post",
                 "body": "Hiya all!"
               },
               {
                 "timestamp": "2020-07-09T01:32:23Z",
                 "title": "Sleepy...",
                 "body": "Catch y'all tomorrow."
               }
             ]
           },
           "favorites": {
             "uint8-numbers": [17, 13, 11, 7, 5, 3],
             "int8-numbers": [-5, -3, -1, 1, 3, 5]
           },
           "stats": {
             "joined": "2020-07-08T12:38:32Z",
             "membership-level": "admin",
             "last-activity": "2021-04-01T02:51:11Z"
           }
         },
         {
           "member-id": "lin",
           "email-address": "lin@example.com",
           "password": "$0$1543",
           "privacy-settings": {
             "hide-network": "true",
             "post-visibility": "followers-only"
           },
            "following": ["joe", "eric", "alice"],
            "stats": {
              "joined": "2020-07-09T12:38:32Z",
              "membership-level": "standard",
              "last-activity": "2021-04-01T02:51:11Z"
            }
          },
          {
            "member-id": "joe",
            "email-address": "joe@example.com",
            "password": "$0$1543",
            "avatar": "BASE64VALUE=",
            "tagline": "Greatness is measured by courage and heart.",
            "privacy-settings": {
              "post-visibility": "unlisted"
            },
            "following": ["bob"],
            "posts": {
              "post": [
                {
                  "timestamp": "2020-10-17T18:02:04Z",
                  "body": "What’s your status?"
                }
              ]
            },
            "stats": {
              "joined": "2020-10-08T12:38:32Z",
              "membership-level": "pro",
              "last-activity": "2021-04-01T02:51:11Z"
            }
          }
        ]
      },
      "example-social:audit-logs": {
        "audit-log": [
          {
            "timestamp": "2020-10-11T06:47:59Z",
            "member-id": "alice",
            "source-ip": "192.168.0.92",
            "request": "POST /groups/group/2043",
            "outcome": true
          },
          {
            "timestamp": "2020-11-01T15:22:01Z",
            "member-id": "bob",
            "source-ip": "192.168.2.16",
            "request": "POST /groups/group/123",
            "outcome": false
```

```
         },
         {
           "timestamp": "2020-12-12T21:00:28Z",
           "member-id": "eric",
           "source-ip": "192.168.254.1",
           "request": "POST /groups/group/10",
           "outcome": true
         },
         {
           "timestamp": "2021-01-03T06:47:59Z",
           "member-id": "alice",
           "source-ip": "192.168.0.92",
           "request": "POST /groups/group/333",
           "outcome": true
         },
         {
           "timestamp": "2021-01-21T10:00:00Z",
           "member-id": "bob",
           "source-ip": "192.168.2.16",
           "request": "POST /groups/group/42",
           "outcome": true
         },
         {
           "timestamp": "2020-02-07T09:06:21Z",
           "member-id": "alice",
           "source-ip": "192.168.0.92",
           "request": "POST /groups/group/1202",
           "outcome": true
         },
         {
           "timestamp": "2020-02-28T02:48:11Z",
           "member-id": "bob",
           "source-ip": "192.168.2.16",
           "request": "POST /groups/group/345",
           "outcome": true
         }
       ]
     }
   }
```

A.3. Example Queries

The following sections are presented in reverse query-parameters processing order. Starting with the simplest (limit) and ending with the most complex (where).

All the vector tests are presented in a protocol-independent manner. JSON is used only for its conciseness.

A.3.1. The "limit" Parameter

Noting that "limit" must be a positive number, the edge condition values are '1', '2', num-elements-1, num-elements, and num-elements+1.

> If '0' were a valid limit value, it would always return an empty result set. Any value greater than or equal to num-elements results the entire result set, same as when "limit" is unspecified.

These vector tests assume the target "/example-social:members/member=alice/favorites/uint8-numbers", which has six values, thus the edge condition "limit" values are: '1', '2', '5', '6', and '7'.

A.3.1.1. limit=1

REQUEST

```
Target: /example-social:members/member=alice/favorites/uint8-numbers
  Pagination Parameters:
    Where:     -
    Sort-by:   -
    Direction: -
    Offset:    -
    Limit:     1
```

RESPONSE

```
{
  "example-social:uint8-numbers": [17],
  "@example-social:uint8-numbers": [
     {
        "ietf-list-pagination:remaining": 5
     }
   ]
}
```

A.3.1.2. limit=2

REQUEST

```
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: -
       Offset:    -
       Limit:     2

   RESPONSE

   {
     "example-social:uint8-numbers": [17, 13],
     "@example-social:uint8-numbers": [
        {
           "ietf-list-pagination:remaining": 4
        }
      ]
   }
```

A.3.1.3. limit=5

```
   REQUEST

   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: -
       Offset:    -
       Limit:     5

   RESPONSE

   {
     "example-social:uint8-numbers": [17, 13, 11, 7, 5],
     "@example-social:uint8-numbers": [
        {
           "ietf-list-pagination:remaining": 1
        }
      ]
   }
```

A.3.1.4. limit=6

```
   REQUEST
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: -
       Offset:    -
       Limit:     6
```

RESPONSE

```
   {
     "example-social:uint8-numbers": [17, 13, 11, 7, 5, 3]
   }
```

A.3.1.5. limit=7

REQUEST

```
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: -
       Offset:    -
       Limit:     7
```

RESPONSE

```
   {
     "example-social:uint8-numbers": [17, 13, 11, 7, 5, 3]
   }
```

A.3.2. The "offset" Parameter

Noting that "offset" must be an unsigned number less than or equal to the num-elements, the edge condition values are ’0’, ’1’, ’2’, num-elements-1, num-elements, and num-elements+1.

These vector tests again assume the target "/example-social:members/member=alice/favorites/uint8-numbers", which has six values, thus the edge condition "limit" values are: ’0’, ’1’, ’2’, ’5’, ’6’, and ’7’.

A.3.2.1. offset=0

REQUEST

```
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: -
       Offset:    0
       Limit:     -
```

RESPONSE

```
   {
     "example-social:uint8-numbers": [17, 13, 11, 7, 5, 3]
   }
```

A.3.2.2. offset=1

REQUEST

```
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: -
       Offset:    1
       Limit:     -
```

RESPONSE

```
   {
     "example-social:uint8-numbers": [13, 11, 7, 5, 3]
   }
```

A.3.2.3. offset=2

REQUEST

```
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: -
       Offset:    2
       Limit:     -
```

RESPONSE

```
   {
     "example-social:uint8-numbers": [11, 7, 5, 3]
   }
```

A.3.2.4. offset=5

REQUEST

```
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: -
       Offset:    5
       Limit:     -
```

RESPONSE

```
   {
     "example-social:uint8-numbers": [3]
   }
```

A.3.2.5. offset=6

REQUEST

```
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: -
       Offset:    6
       Limit:     -
```

RESPONSE

```
   {
     "example-social:uint8-numbers": []
   }
```

A.3.2.6. offset=7

REQUEST

```
Target: /example-social:members/member=alice/favorites/uint8-numbers
  Pagination Parameters:
    Where:     -
    Sort-by:   -
    Direction: -
    Offset:    7
    Limit:     -
```

RESPONSE

ERROR

A.3.3. The "direction" Parameter

Noting that "direction" is an enumeration with two values, the edge condition values are each defined enumeration.

> The value "forwards" is sometimes known as the "default" value, as it produces the same result set as when "direction" is unspecified.

These vector tests again assume the target "/example-social:members/member=alice/favorites/uint8-numbers". The number of elements is relevant to the edge condition values.

> It is notable that "uint8-numbers" is an "ordered-by" user leaf-list. Traversals are over the user-specified order, not the numerically-sorted order, which is what the "sort-by" parameter addresses. If this were an "ordered-by system" leaf-list, then the traversals would be over the system-specified order, again not a numerically-sorted order.

A.3.3.1. direction=forwards

REQUEST

```
Target: /example-social:members/member=alice/favorites/uint8-numbers
  Pagination Parameters:
    Where:     -
    Sort-by:   -
    Direction: forwards
    Offset:    -
    Limit:     -
```

RESPONSE

```
   {
     "example-social:uint8-numbers": [17, 13, 11, 7, 5, 3]
   }
```

A.3.3.2. direction=backwards

REQUEST

```
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   -
       Direction: backwards
       Offset:    -
       Limit:     -
```

RESPONSE

```
   {
     "example-social:uint8-numbers": [3, 5, 7, 11, 13, 17]
   }
```

A.3.4. The "sort-by" Parameter

Noting that the "sort-by" parameter is a node identifier, there is not so much "edge conditions" as there are "interesting conditions". This section provides examples for some interesting conditions.

A.3.4.1. the target node’s type

The section provides three examples, one for a "leaf-list" and two for a "list", with one using a direct descendent and the other using an indirect descendent.

A.3.4.1.1. type is a "leaf-list"

This example illustrates when the target node’s type is a "leaf-list". Note that a single period (i.e., ’.’) is used to represent the nodes to be sorted.

This test again uses the target "/example-social:members/member=alice/favorites/uint8-numbers", which is a leaf-list.

REQUEST

```
   Target: /example-social:members/member=alice/favorites/uint8-numbers
     Pagination Parameters:
       Where:     -
       Sort-by:   .
       Direction: -
       Offset:    -
       Limit:     -
```

RESPONSE

```
   {
     "example-social:uint8-numbers": [3, 5, 7, 11, 13, 17]
   }
```

A.3.4.1.2. type is a "list" and sort-by node is a direct descendent

This example illustrates when the target node’s type is a "list" and a direct descendent is the "sort-by" node.

This vector test uses the target "/example-social:members/member", which is a "list", and the sort-by descendent node "member-id", which is the "key" for the list.

REQUEST

```
   Target: /example-social:members/member
     Pagination Parameters:
       Where:     -
       Sort-by:   member-id
       Direction: -
       Offset:    -
       Limit:     -
```

RESPONSE

> To make the example more understandable, an ellipse (i.e., "...") is used to represent a missing subtree of data.

```
{
  "example-social:member": [
    {
      "member-id": "alice",
      ...
    },
    {
      "member-id": "bob",
      ...
    },
    {
      "member-id": "eric",
      ...
    },
    {
      "member-id": "joe",
      ...
    },
    {
      "member-id": "lin",
      ...
    }
  ]
}
```

A.3.4.1.3. type is a "list" and sort-by node is an indirect descendent

This example illustrates when the target node’s type is a "list" and an indirect descendent is the "sort-by" node.

This vector test uses the target "/example-social:members/member", which is a "list", and the sort-by descendent node "stats/joined", which is a "config false" descendent leaf. Due to "joined" being a "config false" node, this request would have to target the "member" node in the <operational> datastore.

REQUEST

```
Target: /example-social:members/member
  Pagination Parameters:
    Where:     -
    Sort-by:   stats/joined
    Direction: -
    Offset:    -
    Limit:     -
```

RESPONSE

> To make the example more understandable, an elipse (i.e., "...") is used to represent a missing subtree of data.

```
{
  "example-social:member": [
    {
      "member-id": "alice",
      ...
    },
    {
      "member-id": "lin",
      ...
    },
    {
      "member-id": "bob",
      ...
    },
    {
      "member-id": "eric",
      ...
    },
    {
      "member-id": "joe",
      ...
    }
  ]
}
```

A.3.4.2. handling missing entries

The section provides one example for when the "sort-by" node is not present in the data set.

FIXME: need to finish this section...

A.3.5. The "where" Parameter

The "where" is an XPath 1.0 expression, there are numerous edge conditions to consider, e.g., the types of the nodes that are targeted by the expression.

A.3.5.1. match of leaf-list’s values

FIXME

A.3.5.2. match on descendent string containing a substring

This example selects members that have an email address containing "@example.com".

REQUEST

```
Target: /example-social:members/member
  Pagination Parameters:
    Where:     //.[contains (@email-address,'@example.com')]
    Sort-by:   -
    Direction: -
    Offset:    -
    Limit:     -
```

RESPONSE

> To make the example more understandable, an elipse (i.e., "...") is used to represent a missing subtree of data.

```
{
  "example-social:member": [
    {
      "member-id": "bob",
      ...
    },
    {
      "member-id": "eric",
      ...
    },
    {
      "member-id": "alice",
      ...
    },
    {
      "member-id": "joe",
      ...
    },
    {
      "member-id": "lin",
      ...
    }
  ]
}
```

A.3.5.3. match on decendent timestamp starting with a substring

This example selects members that have a posting whose timestamp begins with the string "2020".

REQUEST

```
Target: /example-social:members/member
  Pagination Parameters:
    Where:     //posts//post[starts-with(@timestamp,'2020')]
    Sort-by:   -
    Direction: -
    Offset:    -
    Limit:     -
```

RESPONSE

> To make the example more understandable, an elipse (i.e., "...") is used to represent a missing subtree of data.

```
{
  "example-social:member": [
    {
      "member-id": "bob",
      ...
    },
    {
      "member-id": "eric",
      ...
    },
    {
      "member-id": "alice",
      ...
    },
    {
      "member-id": "joe",
      ...
    }
  ]
}
```

A.3.6. The "sublist-limit" Parameter

The "sublist-limit" parameter may be used on any target node.

A.3.6.1. target is a list entry

This example uses the target node '/example-social:members/member=alice' in the <intended> datastore.

> The target node is a specific list entry/element node, not the YANG "list" node.

This example sets the sublist-limit value '1', which returns just the first entry for all descendent lists and leaf-lists.

Note that, in the response, the "remaining" metadata value is set on the first element of each descendent list and leaf-list having more than one value.

REQUEST

```
  Datastore: <intended>
  Target: /example-social:members/member=alice
  Sublist-limit: 1
  Pagination Parameters:
    Where:     -
    Sort-by:   -
    Direction: -
    Offset:    -
    Limit:     -
```

RESPONSE

```
   {
     "example-social:member": [
       {
         "member-id": "alice",
         "email-address": "alice@example.com",
         "password": "$0$1543",
         "avatar": "BASE64VALUE=",
         "tagline": "Every day is a new day",
         "privacy-settings": {
           "hide-network": "false",
           "post-visibility": "public"
         },
         "following": ["bob"],
         "@following": [
           {
             "ietf-list-pagination:remaining": "2"
           }
         ],
         "posts": {
           "post": [
             {
               "@": {
                 "ietf-list-pagination:remaining": "1"
               },
               "timestamp": "2020-07-08T13:12:45Z",
               "title": "My first post",
               "body": "Hiya all!"
             }
           ]
         },
         "favorites": {
           "uint8-numbers": [17],
           "int8-numbers": [-5],
           "@uint8-numbers": [
             {
               "ietf-list-pagination:remaining": "5"
             }
           ],
           "@int8-numbers": [
             {
               "ietf-list-pagination:remaining": "5"
             }
           ]
         }
       }
     ]
   }
```

A.3.6.2. target is a datastore

This example uses the target node <intended>.

This example sets the sublist-limit value '1', which returns just the first entry for all descendent lists and leaf-lists.

Note that, in the response, the "remaining" metadata value is set on the first element of each descendent list and leaf-list having more than one value.

REQUEST

```
  Datastore: <intended>
  Target: /
  Sublist-limit: 1
  Pagination Parameters:
    Where:     -
    Sort-by:   -
    Direction: -
    Offset:    -
    Limit:     -
```

RESPONSE

```
   {
     "example-social:members": {
       "member": [
         {
           "@": {
             "ietf-list-pagination:remaining": "4"
           },
           "member-id": "bob",
           "email-address": "bob@example.com",
           "password": "$0$1543",
           "avatar": "BASE64VALUE=",
           "tagline": "Here and now, like never before.",
           "posts": {
             "post": [
               {
                 "@": {
                   "ietf-list-pagination:remaining": "2"
                 },
                 "timestamp": "2020-08-14T03:32:25Z",
                 "body": "Just got in."
               }
             ]
           },
           "favorites": {
             "decimal64-numbers": ["3.14159"],
             "@decimal64-numbers": [
               {
                 "ietf-list-pagination:remaining": "1"
               }
             ]
           }
         }
       ]
     }
   }
```

A.3.7. Combinations of Parameters

A.3.7.1. All six parameters at once

REQUEST

```
     Datastore: <operational>
     Target: /example-social:members/member
     Sublist-limit: 1
     Pagination Parameters:
       Where:     //stats//joined[starts-with(@timestamp,'2020')]
       Sort-by:   member-id
       Direction: backwards
       Offset:    2
       Limit:     2
```

RESPONSE

```
   {
     "example-social:member": [
       {
         "@": {
           "ietf-list-pagination:remaining": "1"
         },
         "member-id": "eric",
         "email-address": "eric@example.com",
         "password": "$0$1543",
         "avatar": "BASE64VALUE=",
         "tagline": "Go to bed with dreams; wake up with a purpose.",
         "following": ["alice"],
         "posts": {
           "post": [
             {
               "timestamp": "2020-09-17T18:02:04Z",
               "title": "Son, brother, husband, father",
               "body": "What's your story?"
             }
           ]
         },
         "favorites": {
           "bits": ["two"],
           "@bits": [
             {
               "ietf-list-pagination:remaining": "2"
             }
           ]
         },
         "stats": {
           "joined": "2020-09-17T19:38:32Z",
           "membership-level": "pro",
           "last-activity": "2020-09-17T18:02:04Z"
         }
       },
       {
         "member-id": "bob",
         "email-address": "bob@example.com",
         "password": "$0$1543",
         "avatar": "BASE64VALUE=",
         "tagline": "Here and now, like never before.",
         "posts": {
           "post": [
             {
               "@": {
                 "ietf-list-pagination:remaining": "2"
               },
               "timestamp": "2020-08-14T03:32:25Z",
               "body": "Just got in."
             }
           ]
         },
         "favorites": {
           "decimal64-numbers": ["3.14159"],
           "@decimal64-numbers": [
             {
               "ietf-list-pagination:remaining": "1"
             }
           ]
         },
         "stats": {
           "joined": "2020-08-14T03:30:00Z",
           "membership-level": "standard",
           "last-activity": "2020-08-14T03:34:30Z"
         }
       }
     }
   }
```

Acknowledgements

The authors would like to thank the following for lively discussions on list (ordered by first name): Andy Bierman, Martin Bjoerklund, and Robert Varga.

Authors' Addresses

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Qin Wu
Huawei Technologies

Email: bill.wu@huawei.com

Olof Hagsand
Netgate

Email: olof@hagsand.se

Hongwei Li
Hewlett Packard Enterprise

Email: flycoolman@gmail.com

Per Andersson
Cisco Systems

Email: perander@cisco.com

NETCONF Working Group   K. Watsen
Internet-Draft   Watsen Networks
Intended status: Standards Track   Q. Wu
Expires: 28 April 2022   Huawei
                                                              O. Hagsand
                                                                 Netgate
                                                                   H. Li
                                                                     HPE
                                                             P. Andersson
                                                           Cisco Systems
                                                         25 October 2021

NETCONF Extensions to Support List Pagination
draft-wwlh-netconf-list-pagination-nc-02

Abstract

This document defines a mapping of the list pagination mechanism defined in [I-D.wwlh-netconf-list-pagination] to NETCONF [RFC6241].

This document updates [RFC6241], to augment the <get> and <get-config> "rpc" statements, and [RFC8526], to augment the <get-data> "rpc" statement, to define input parameters necessary for list pagination.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a mapping of the list pagination mechanism defined in [I-D.wwlh-netconf-list-pagination] to NETCONF [RFC6241].

This document updates [RFC6241] and [RFC8526], as described in Section 2.

While the pagination mechanism defined in this document is designed for the NETCONF protocol [RFC6241], the augmented RPCs MAY be used by the RESTCONF protocol [RFC8040] if the RESTCONF server implements the "ietf-list-pagination-nc" module.

The YANG data model in this document conforms to the Network Management Datastore Architecture defined in [RFC8342]

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.

1.2. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. Updates to NETCONF operations

2.1. Updates to RFC 6241

The <get> and <get-config> rpc statements are augmented to accept additional input parameters, as described in Section 3.

2.2. Updates to RFC 8526

The <get-data> rpc statement is augmented to accept additional input parameters, as described in in Section 3.

3. List Pagination for NETCONF

In order for NETCONF to support [I-D.wwlh-netconf-list-pagination], this document extends the operations <get>, <get-config> and <get-data> to include additional input parameters and output annotations.

The updated operations accept a content filter parameter, similar to the "filter" parameter of <get-config>, but includes nodes for "list" and "leaf-list" filtering.

The content filter parameter is used to specify the YANG list or leaf-list that is to be retrieved. This must be a path expression used to represent a list or leaf-list data node.

The following tree diagram [RFC8340] illustrates the "ietf-netconf-list-pagination" module:

```
   module: ietf-list-pagination-nc

     augment /nc:get/nc:input:
       +---w list-pagination
          +---w where?           union
          +---w sort-by?         union
          +---w direction?       enumeration
          +---w offset?          uint32
          +---w limit?           union
          +---w sublist-limit?   union
     augment /nc:get-config/nc:input:
       +---w list-pagination
          +---w where?           union
          +---w sort-by?         union
          +---w direction?       enumeration
          +---w offset?          uint32
          +---w limit?           union
          +---w sublist-limit?   union
     augment /ncds:get-data/ncds:input:
       +---w list-pagination
          +---w where?           union
          +---w sort-by?         union
          +---w direction?       enumeration
          +---w offset?          uint32
          +---w limit?           union
          +---w sublist-limit?   union
```

Comments:

* This module augments three NETCONF "rpc" statements: get, get-config, and get-data.

* The "get" and "get-config" augments are against the YANG module defined in [RFC6241]. The "get-data" augment is against the YANG module defined in [RFC8526].

4. Error Reporting

When an input query parameter is supplied with an erroneous value, an <rpc-error> MUST be returned containing the error-type value "application", the error-tag value "invalid-value", and MAY include the error-severity value "error". Additionally the error-app-tag SHOULD be set containing query parameter specific error value.

4.1. The "offset" Query Parameter

If the "offset" query parameter value supplied is larger then the number of instances in the list or leaf-list target resource, the <rpc-error> MUST contain error-app-tag with value "offset-out-of-range".

5. YANG Module for List Pagination in NETCONF

The "ietf-netconf-list-pagination-nc" module defines conceptual definitions within groupings, which are not meant to be implemented as datastore contents by a server.

This module has normative references to [RFC6241], [RFC6243], [RFC6991], and [RFC8342].

```
<CODE BEGINS> file "ietf-list-pagination-nc@2021-10-25.yang"

module ietf-list-pagination-nc {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-list-pagination-nc";
  prefix lpgnc;

  import ietf-netconf {
    prefix nc;
    reference
      "RFC 6241: Network Configuration Protocol (NETCONF)";
  }

  import ietf-netconf-nmda {
    prefix ncds;
    reference
      "RFC 8526: NETCONF Extensions to Support the
                 Network Management Datastore Architecture";
  }

  import ietf-list-pagination {
    prefix lp;
    reference
      "RFC XXXX: List Pagination for YANG-driven Protocols";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";

  contact
      "WG Web:   <http://tools.ietf.org/wg/netconf/>
       WG List:  <mailto:netconf@ietf.org>";
```

```
     description
       "This module augments the <get>, <get-config>, and <get-data>
        'rpc' statements to support list pagination.

        Copyright (c) 2021 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Simplified
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX
        (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
        itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

     revision 2021-10-25 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: NETCONF Extensions to Support List Pagination";
     }

     grouping pagination-parameters {
       description "A grouping for list pagination parameters.";
       container list-pagination {
         description "List pagination parameters.";
         uses lp:where-param-grouping;
         uses lp:sort-by-param-grouping;
         uses lp:direction-param-grouping;
         uses lp:offset-param-grouping;
         uses lp:limit-param-grouping;
         uses lp:sublist-limit-param-grouping;
       }
     }

     augment "/nc:get/nc:input" {
       description
         "Allow the 'get' operation to use content filter
```

```
          parameter for specifying the YANG list or leaf-list
          that is to be retrieved";
       uses pagination-parameters;
     }

     augment "/nc:get-config/nc:input" {
       description
         "Allow the 'get-config' operation to use content filter
          parameter for specifying the YANG list or leaf-list
          that is to be retrieved";
       uses pagination-parameters;
     }

     augment "/ncds:get-data/ncds:input" {
       description
         "Allow the 'get-data' operation to use content filter
          parameter for specifying the YANG list or leaf-list
          that is to be retrieved";
       uses pagination-parameters;
     }
   }

   <CODE ENDS>
```

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688] maintained at https://www.iana.org/assignments/xml-registry/xml-registry.xhtml#ns. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-list-pagination-nc
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

6.2. The "YANG Module Names" Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020] maintained at https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml. Following the format defined in [RFC6020], the below registration is requested:

```
   name: ietf-list-pagination-nc
   namespace: urn:ietf:params:xml:ns:yang:ietf-list-pagination-nc
   prefix: pgnc
   RFC: XXXX
```

7. Security Considerations

7.1. The "ietf-netconf-list-pagination" YANG Module

The YANG module defined in this document extends the base operations for NETCONF [RFC6241] and RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

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

The security considerations for the base NETCONF protocol operations (see Section 9 of [RFC6241] apply to the new <get-list-pagination> RPC operations defined in this document.

8. References

8.1. Normative References

[I-D.wwlh-netconf-list-pagination]
"List Pagination...", <FIXME>.

[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>.

[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>.

[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>.

[RFC6243] Bierman, A. and B. Lengyel, "With-defaults Capability for NETCONF", RFC 6243, DOI 10.17487/RFC6243, June 2011, <https://www.rfc-editor.org/info/rfc6243>.

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

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8526] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "NETCONF Extensions to Support the Network Management Datastore Architecture", RFC 8526, DOI 10.17487/RFC8526, March 2019, <https://www.rfc-editor.org/info/rfc8526>.

8.2. Informative References

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

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

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

Appendix A. Open Issues

Cursors (i.e.,stable result sets) are related to the topic of dynamic changing lists between two queries. How cursors can be supported using "feature"?

Appendix B. Example YANG Module

The examples within this document use the "example-social" YANG module defined in Appendix A.1 of [I-D.wwlh-netconf-list-pagination].

Appendix C. Example Data Set

The Example Data Set used by the examples is defined in Appendix A.2 of [I-D.wwlh-netconf-list-pagination].

Appendix D. Example Queries

D.1. List pagination with all query parameters

This example mimics that Appendix A.3.7 of [I-D.wwlh-netconf-list-pagination].

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

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="42">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="xpath" select="/es:members/es:member"
      xmlns:es="http://example.com/ns/example-social"/>
      <list-pagination
        xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-list-paginat\
ion">true</list-pagination>
      <where>//stats//joined[starts-with(@timestamp,'2020')]</where>
      <sort-by>timestamp</sort-by>
      <direction>backwards</direction>
      <offset>2</offset>
      <limit>2</limit>
      <sublist-limit>1</sublist-limit>
    </filter>
  </get-config>
</rpc>
```

Response from the NETCONF server:

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

<lp:xml-list xmlns:lp="urn:ietf:params:xml:ns:yang:ietf-restconf-lis\
t-pagination"
  xmlns="http://example.com/ns/example-social">
  <member lp:remaining="1">
    <member-id>eric</member-id>
    <email-address>eric@example.com</email-address>
    <password>$0$1543</password>
    <avatar>BASE64VALUE=</avatar>
    <tagline>Go to bed with dreams; wake up with a purpose.</tagline>
    <following>alice</following>
    <posts>
      <post>
        <timestamp>2020-09-17T18:02:04Z</timestamp>
        <title>Son, brother, husband, father</title>
        <body>What's your story?</body>
      </post>
    </posts>
    <favorites>
      <bits lp:remaining="2">two</bits>
    </favorites>
    <stats>
      <joined>2020-09-17T19:38:32Z</joined>
      <membership-level>pro</membership-level>
      <last-activity>2020-09-17T18:02:04Z</last-activity>
    </stats>
  </member>
  <member lp:remaining="1">
    <member-id>bob</member-id>
    <email-address>bob@example.com</email-address>
    <password>$0$1543</password>
    <avatar>BASE64VALUE=</avatar>
    <tagline>Here and now, like never before.</tagline>
    <posts>
      <post lp:remaining="2">
        <timestamp>2020-08-14T03:32:25Z</timestamp>
        <body>Just got in.</body>
      </post>
    </posts>
    <favorites>
      <decimal64-numbers lp:remaining="1">3.14159</bits>
    </favorites>
    <stats>
      <joined>2020-08-14T03:30:00Z</joined>
      <membership-level>standard</membership-level>
```

```
         <last-activity>2020-08-14T03:34:30Z</last-activity>
       </stats>
     </member>
   </lp:xml-list>
```

Acknowledgements

This work has benefited from the discussions of RESTCONF resource collection over the years, in particular, [I-D.ietf-netconf-restconf-collection] which provides enhanced filtering features for the retrieval of data nodes with the GET method and [I-D.zheng-netconf-fragmentation] which document large size data handling challenge. The authors would like to thank the following for lively discussions on list:

Andy Bierman Martin Björklund Robert Varga

Authors' Addresses

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: bill.wu@huawei.com

Olof Hagsand
Netgate

Email: olof@hagsand.se

Hongwei Li
HPE

Email: flycoolman@gmail.com

Per Andersson
Cisco Systems

Email: perander@cisco.com

```
NETCONF Working Group                                          K. Watsen
Internet-Draft                                           Watsen Networks
Updates: 8040 (if approved)                                        Q. Wu
Intended status: Standards Track                     Huawei Technologies
Expires: 28 April 2022                                        O. Hagsand
                                                                 Netgate
                                                                   H. Li
                                             Hewlett Packard Enterprise
                                                            P. Andersson
                                                           Cisco Systems
                                                         25 October 2021
```

RESTCONF Extensions to Support List Pagination
draft-wwlh-netconf-list-pagination-rc-02

Abstract

This document defines a mapping of the list pagination mechanism defined in [I-D.wwlh-netconf-list-pagination] to RESTCONF [RFC8040].

This document updates RFC 8040, to declare "list" and "leaf-list" as valid resource targets for the RESTCONF GET and DELETE operations, to define GET query parameters necessary for list pagination, and to define a media-type for XML-based lists.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document defines a mapping of the list pagination mechanism defined in [I-D.wwlh-netconf-list-pagination] to RESTCONF [RFC8040].

This document updates RFC 8040, as described in Section 2.

Declaring "list" and "leaf-list" as valid resource targets for the GET operation is necessary for list pagination. Declaring these nodes as valid resource targets for the DELETE operation merely completes the solution for RESTCONF.

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.

1.2. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. Updates to RFC 8040

2.1. Resource Targets

This document extends Section 3.5 of [RFC8040] to add "list" and "leaf-list" nodes (not just their entries) as valid data resources for the "GET" and "DELETE" operations.

2.2. Media Type

This document extends Section 3.2 of [RFC8040] to add a new media type, "application/yang-data+xml-list", to encode "list" and "leaf-list" nodes in XML.

The "application/yang-data+xml-list" media-type defines a pseudo top-level element called "xml-list" that is used to wrap the response set, thus ensuring that a single top-level element is returned for the XML encoding", as required by Section 4.3 of [RFC8040].

For JSON, the existing "application/yang-data+json" media type is sufficient, as the JSON format has built-in support for encoding arrays.

The "application/yang-data+xml-list" media type is registered in Section 3.2.1.

2.3. Query Parameters

This document extends Section 4.8 of [RFC8040] to add new query parameters "limit", "offset", "direction", "sort-by", "where", and "sublist-list".

These six query parameters correspond to those defined in Sections 3.1 and 3.2 in [I-D.wwlh-netconf-list-pagination].

Name	Methods	Description
limit	GET, HEAD	Limits the number of entries returned. If not specified, the number of entries that may be returned is unbounded.
offset	GET, HEAD	Indicates the number of entries in the result set that should the skipped over when preparing the response. If not specified, then no entries in the result set are skipped.
direction	GET, HEAD	Indicates the direction that the result set is to be traversed. If not specified, then the result set is traversed in the "forwards" direction.
sort-by	GET, HEAD	Indicates the node name that the result set should be sorted by. If not specified, then the result set's default order is used, per YANG's "ordered-by" statement.
where	GET, HEAD	Specifies a filter expression that result set entries must match. If not specified, then no entries are filtered from the result set.
sublist-limit	GET, HEAD	Limits the number of entries returned returned for descendent lists and leaf-lists. If not specified, the number of entries that may be returned is unbounded.

For all of the query parameters, the query parameter is only allowed for the GET and HEAD methods on "list" and "leaf-list" data resources. A "400 Bad Request" status-line MUST be returned if used with any other method or resource type. The error-tag value "operation-not-supported" is used in this case.

Per the conformance defined in Section 3.1 of [I-D.wwlh-netconf-list-pagination], all of these parameters MUST be supported for all lists and leaf-lists, but servers MAY disable the support for some or all "config false" lists, as described in Section 3.3 of [I-D.wwlh-netconf-list-pagination].

2.3.1. The "limit" Query Parameter

The "limit" query parameter corresponds to the "limit" parameter defined in Section 3.1.5 of [I-D.wwlh-netconf-list-pagination].

If the limit value is invalid, then a "400 Bad Request" status-line MUST be returned with the error-type value "application" and error-tag value "invalid-value".

2.3.2. The "offset" Query Parameter

The "offset" query parameter corresponds to the "offset" parameter defined in Section 3.1.4 of [I-D.wwlh-netconf-list-pagination].

If the offset value is invalid, a "400 Bad Request" status-line MUST be returned with the error-type value "application" and error-tag value "invalid-value".

If the offset value exceeds the number of entries in the working result set, then a "416 Range Not Satisfiable" status-line MUST be returned with the error-type value "application", error-tag value "invalid-value", and SHOULD also include the "offset-out-of-range" identity as error-app-tag value.

2.3.3. The "direction" Query Parameter

The "direction" query parameter corresponds to the "direction" parameter defined in Section 3.1.3 of [I-D.wwlh-netconf-list-pagination].

If the direction value is invalid, then a "400 Bad Request" status-line MUST be returned with the error-type value "application" and error-tag value "invalid-value".

2.3.4. The "sort-by" Query Parameter

The "sort-by" query parameter corresponds to the "sort-by" parameter defined in Section 3.1.2 of [I-D.wwlh-netconf-list-pagination].

If the specified node identifier is invalid, then a "400 Bad Request" status-line MUST be returned with the error-type value "application" and error-tag value "invalid-value".

2.3.5. The "where" Query Parameter

The "where" query parameter corresponds to the "where" parameter defined in Section 3.1.1 of [I-D.wwlh-netconf-list-pagination].

If the specified XPath expression is invalid, then a "400 Bad Request" status-line MUST be returned with the error-type value "application" and error-tag value "invalid-value".

2.3.6. The "sublist-limit" Query Parameter

The "sublist-limit" query parameter corresponds to the "sublist-limit" parameter defined in Section 3.2.1 of [I-D.wwlh-netconf-list-pagination].

If the sumlist-limit value is invalid, then a "400 Bad Request" status-line MUST be returned with the error-type value "application" and error-tag value "invalid-value".

3. IANA Considerations

3.1. The "RESTCONF Capability URNs" Registry

This document registers six capabilities in the RESTCONF Capability URNs [RFC8040] maintained at https://www.iana.org/assignments/restconf-capability-urns/restconf-capability-urns.xhtml. Following the instructions defined in Section 11.4 of [RFC8040], the below registrations are requested:

All the registrations are to use this document (RFC XXXX) for the "Reference" value.

Index	Capability Identifier
:limit	urn:ietf:params:restconf:capability:limit:1.0
:offset	urn:ietf:params:restconf:capability:offset:1.0
:direction	urn:ietf:params:restconf:capability:direction:1.0
:sort-by	urn:ietf:params:restconf:capability:sort-by:1.0
:where	urn:ietf:params:restconf:capability:where:1.0
:sublist-limit	urn:ietf:params:restconf:capability:sublist-limit:1.0

3.2. The "Media Types" Registry

This document registers one media type in the "application" subregistry of the Media Types registry [RFC6838] [RFC4855] maintained at https://www.iana.org/assignments/media-types/media-types.xhtml#application. Following the format defined in [RFC4855], the below registration is requested:

3.2.1. Media Type "application/yang-data+xml-list"

Type name: application

Subtype name: yang-data+xml-list

Required parameters: None

Optional parameters: None

Encoding considerations: 8-bit
Each conceptual YANG data node is encoded according to the XML Encoding Rules and Canonical Format for the specific YANG data node type defined in [RFC7950].

Security considerations: Security considerations related to the generation and consumption of RESTCONF messages are discussed in Section 12 of RFC 8040. Additional security considerations are specific to the semantics of particular YANG data models. Each YANG module is expected to specify security considerations for the YANG data defined in that module.

Interoperability considerations: RFC XXXX specifies the format of conforming messages and the interpretation thereof.

Published specification: RFC XXXX

Applications that use this media type: Instance document data parsers used within a protocol or automation tool that utilize the YANG Patch data structure.

Fragment identifier considerations: Fragment identifiers for this type are not defined. All YANG data nodes are accessible as resources using the path in the request URI.

Additional information:

Deprecated alias names for this type: N/A
Magic number(s): N/A
File extension(s): None
Macintosh file type code(s): "TEXT"

Person & email address to contact for further information: See the Authors' Addresses section of RFC XXXX.

Intended usage: COMMON

Restrictions on usage: N/A

Author: See the Authors' Addresses section of RFC XXXX.

Change controller: Internet Engineering Task Force (mailto:iesg@ietf.org).

Provisional registration? (standards tree only): no

4. Security Considerations

This document introduces protocol operations for paging through data already provided by the RESTCONF protocol, and hence does not introduce any new security considerations.

This document does not define a YANG module and hence there are no data modeling considerations beyond those discussed in [I-D.wwlh-netconf-list-pagination].

5. References

5.1. Normative References

[I-D.wwlh-netconf-list-pagination]
"List Pagination...", <FIXME>.

[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>.

[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>.

5.2. Informative References

[I-D.ietf-netconf-restconf-collection]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Collection Resource", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-collection-00, 30 January 2015, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-collection-00>.

[RFC4855] Casner, S., "Media Type Registration of RTP Payload Formats", RFC 4855, DOI 10.17487/RFC4855, February 2007, <https://www.rfc-editor.org/info/rfc4855>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

Appendix A. Example YANG Module

The examples within this document use the "example-social" YANG module defined in Appendix A.1 of [I-D.wwlh-netconf-list-pagination].

Appendix B. Example Data Set

The Example Data Set used by the examples is defined in Appendix A.2 of [I-D.wwlh-netconf-list-pagination].

Appendix C. Example Queries

C.1. List pagination with all query parameters

This example mimics that Appendix A.3.7 of [I-D.wwlh-netconf-list-pagination].

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

GET /restconf/ds/ietf-datastores:running/example-social:members/memb\
er?where=//stats//joined[starts-with(@timestamp,'2020')]&sort-by=tim\
estamp&direction=backwards&offset=2&limit=2&sublist-limit=1  HTTP/1.1
Host: example.com
Accept: application/yang-data+xml-list
```

Response from the RESTCONF server:

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

HTTP/1.1 200 OK
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
Last-Modified: Thu, 26 Jan 2017 20:55:30 GMT
Content-Type: application/yang-data+xml-list

<lp:xml-list xmlns:lp="urn:ietf:params:xml:ns:yang:ietf-restconf-lis\
t-pagination"
  xmlns="http://example.com/ns/example-social">
  <member lp:remaining="1">
    <member-id>eric</member-id>
    <email-address>eric@example.com</email-address>
    <password>$0$1543</password>
    <avatar>BASE64VALUE=</avatar>
    <tagline>Go to bed with dreams; wake up with a purpose.</tagline>
    <following>alice</following>
    <posts>
      <post>
        <timestamp>2020-09-17T18:02:04Z</timestamp>
        <title>Son, brother, husband, father</title>
        <body>What's your story?</body>
      </post>
    </posts>
    <favorites>
      <bits lp:remaining="2">two</bits>
    </favorites>
    <stats>
      <joined>2020-09-17T19:38:32Z</joined>
      <membership-level>pro</membership-level>
      <last-activity>2020-09-17T18:02:04Z</last-activity>
    </stats>
  </member>
  <member lp:remaining="1">
    <member-id>bob</member-id>
    <email-address>bob@example.com</email-address>
    <password>$0$1543</password>
```

```
       <avatar>BASE64VALUE=</avatar>
       <tagline>Here and now, like never before.</tagline>
       <posts>
         <post lp:remaining="2">
           <timestamp>2020-08-14T03:32:25Z</timestamp>
           <body>Just got in.</body>
         </post>
       </posts>
       <favorites>
         <decimal64-numbers lp:remaining="1">3.14159</bits>
       </favorites>
       <stats>
         <joined>2020-08-14T03:30:00Z</joined>
         <membership-level>standard</membership-level>
         <last-activity>2020-08-14T03:34:30Z</last-activity>
       </stats>
     </member>
   </lp:xml-list>
```

C.2. Deletion of a list

This example illustrates using a "list" as the DELETE target.

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

DELETE /restconf/ds/ietf-datastores:running/example-social:members/m\
ember=bob/favorites/decimal64-numbers HTTP/1.1
Host: example.com
Accept: application/yang-data+xml
```

Response from the RESTCONF server:

```
HTTP/1.1 204 No Content
Date: Thu, 26 Jan 2017 20:56:30 GMT
Server: example-server
```

Acknowledgements

This work has benefited from the discussions of restconf resource collection over the years, in particular, [I-D.ietf-netconf-restconf-collection]. The authors additionally thank the following for lively discussions on list (ordered by first name): Andy Bierman, Martin Bjoerklund, and Robert Varga

Authors' Addresses

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net

Qin Wu
Huawei Technologies
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China

Email: bill.wu@huawei.com

Olof Hagsand
Netgate

Email: olof@hagsand.se

Hongwei Li
Hewlett Packard Enterprise

Email: flycoolman@gmail.com

Per Andersson
Cisco Systems

Email: perander@cisco.com