NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 6 December 2026

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

4 June 2026

List Pagination for YANG-driven Protocols
draft-ietf-netconf-list-pagination-12

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.

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 (e.g., long loading time, memory overconsuming, or crash) 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]. How the NETCONF and RESTCONF protocols support list pagination or protocols mapping is described in [I-D.ietf-netconf-list-pagination-nc] and [I-D.ietf-netconf-list-pagination-rc], respectively.

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 in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (per Section 9.8 of [RFC7950]). This placeholder value is used because 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 defines the "sort" feature, annotations and identities for protocol signaling, a grouping for the list paginaton query parameters, and also augments 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.

Furthermore the solution is intended to leverage underlying database system capabilities, e.g. fast lookups relying on indexes, using efficient built-in selection and pagination functions. However, since there are many different database systems and configurations, the solution defines a common subset of functionality broadly available. It is also possible that a datastore’s underlying database system is federated, i.e. consists of several different database systems to provide one datastore. In order to form a general solution, the possibility to configure and tune what components are available is presented.

A secondary aspect of this solution entails a client sending a query parameter to a server guiding how descendant 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 in 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 a combination of the "direction" parameter (see Section 3.1.4), with either the "offset" parameter (see Section 3.1.5) or the "cursor" parameter (see Section 3.1.6), and lastly the "limit" parameter (see Section 3.1.7).

The sorting can furthermore be configured with a locale for sorting. This is done by setting the "locale" parameter (see Section 3.1.3).

The feature "sort" is used to indicate support for the query parameters "locale" and "sort-by".

3.1.1. The "where" Query Parameter

Description
: The "where" query parameter specifies a filter expression that result-set entries must match. If the string "unfiltered" is supplied, no entries are filtered from the working result-set.

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

Allowed Values
: The allowed values are XPath 1.0 expressions. The XPath context follows Section 6.4.1 of [RFC7950] and, in addition, the context node is the target of the query, the accessible tree can be specific entry of "config true" lists and leaf-lists or "config false" lists and leaf-lists. Details (such as prefix bindings) are further defined at the protocol level, see [I-D.ietf-netconf-list-pagination-nc] and [I-D.ietf-netconf-list-pagination-rc]. If an XPath expression references a node identifier that does not exist in the schema, or use undefined prefixes, or is optional or conditional in the schema, the filter evaluates to false and nothing is filtered from the result-set. It is an error if the XPath expression references constrained "config false" lists and leaf-lists (see Section 3.3), if the node identifier does not point to a node having the "indexed" leaf set to "true" (see Section 3.3.2).

Conformance
: When the "where" feature is enabled, 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. The supplied expression is allowed to be of arbitrary complexity. 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. Clients are allowed to request sorting by any data node within the result-set. It is an error if the specified node identifier does not exist in the schema, 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" leaf set to "true" (see Section 3.3.2). If the specified node identifier is optional or conditional in the schema, the default strategy should be applied to entries without the sort-by value, i.e., place entries without the sort-by value after all entries that have a value.

Conformance
: When the "sort-by" feature is enabled, the "sort-by" query parameter MUST be supported for all "config true" lists and leaf-lists and SHOULD be supported for all "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 "locale" Query Parameter

Description
: The "locale" query parameter indicates what locale is used when sorting the result-set. Note that the "locale" query parameter is invalid to supply without also supplying the "sort-by" query parameter. If a query supplies "locale" and not "sort-by", error-type application and error-tag "invalid-value" is returned.

Default Value
: If this query parameter is unspecified, it is up to the server to select a locale for sorts. How the server chooses the locale used is out of scope for this document. The result-set includes the locale used by the server for sorts with a metadata value [RFC7952] called "locale".

Allowed Values
: The format is a free form string but SHOULD follow the language sub-tag format defined in [RFC5646]. An example is 'sv_SE'. If a supplied locale is unknown to the server, the "locale-unavailable" SHOULD be produced in the error-app-tag in the error output. Note that all locales are assumed to be UTF-8, since character encoding for YANG strings and all known YANG modelled encodings and protocols are required to be UTF-8 [RFC6241] [RFC7950] [RFC7951]

[RFC8040]. A server MUST accept a known encoding with or without trailing ".UTF-8" and MAY emit an encoding with or without trailing ".UTF-8". This means a server must handle both e.g. "sv_SE" and "sv_SE.UTF-8" equally as input, and chooses how to emit used locale as output.

Conformance
When the "sort-by" parameter is specified, the "locale" query parameter MUST be supported for all "config true" lists and leaf-lists 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.

3.1.4. 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.

3.1.5. 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. The "offset" query parameter MUST not be used together with "cursor" query parameter

Default Value
: If this query parameter is unspecified, then no entries in the result-set are skipped, the 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-app-tag in the error output when this occurs.

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

3.1.6. The "cursor" Query Parameter

Description
: The "cursor" query parameter indicates where to start the working result-set (i.e., after the "direction" parameter has been applied), the elements before the cursor are skipped over when preparing the response. Furthermore, a result set constrained with the "limit" query parameter includes metadata values [RFC7952] called "next" and "previous", which contains cursor values to the next and previous result-sets. These next and previous cursor values are opaque index values for the underlying system’s database, e.g. a key or other information needed to efficiently access the selected result-set. These "next" and "previous" metadata values work as Hypermedia as the Engine of Application State (HATEOAS) links [REST-Dissertation]. This means that the server does not keep any stateful information about the "next" and "previous" cursor or the current page. Due to their ephemeral nature, cursor values are never cached.

Default Value
: If this query parameter is unspecified, then no entries in the result-set are skipped.

Allowed Values
: The allowed values are opaque encoded positions interpreted by the server to index an element in the list, e.g. a list key or other information to efficiently access the selected result-set. It is an error to supply an unknown cursor value for the working result-set, and the "cursor-not-found" identity SHOULD be produced in the error-app-tag in the error output when this occurs.

Conformance

The "cursor" query parameter MUST be supported for all "config true" lists and SHOULD be supported for all "config false" lists. As the "cursor" query parameter support for "config false" lists is optional, the support MUST be signaled by the server per list.

Servers indicate that they support the "cursor" query parameter for a "config false" list node by having the "cursor-supported" leaf set to "true" in the "per-node-capabilities" node in the "ietf-system-capabilities" model.

Since leaf-lists might not have any unique values that can be indexed, the "cursor" query parameter is not relevant for leaf-lists. Consider the following leaf-list [1,1,2,3,5], which contains elements without uniquely indexable values. It would be possible to use the position, but then the solution would be equal to using the "offset" query parameter.

3.1.7. 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. If the string "unbounded" is supplied, there is no limit imposed on the result-set.

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

Allowed Values
: The allowed values are unsigned 32 bit integers greater than or equal to 1, or the string "unbounded".

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 descendant lists and leaf-lists.

  Any descendant 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 descendant 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, translating 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 servers must already 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 [RFC9196].

Specifically, the "ietf-list-pagination" module (see Section 4) augments a boolean 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 set to "true" (see Section 3.3.2).

* For lists only, node-identifiers used in "where" expressions and "sort-by" filters MUST NOT descend past any descendant 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 a boolean leaf node called "indexed" into the "per-node-capabilities" node defined in the "ietf-system-capabilities" module (see [RFC9196]).

When a "list" or "leaf-list" node has the "constrained" leaf, only nodes having the "indexed" node set to "true" may be used in "where" and/or "sort-by" expressions. If no nodes have the "indexed" leaf set to "true", when the "constrained" leaf is set to "true", 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?        boolean
    +--ro indexed?            boolean
    +--ro cursor-supported?   boolean
```

Comments:

* As shown, this module augments three optional leafs into the "per-node-capabilities" 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 3.1.7) or "sublist-limit" parameters (Appendix A.3.8).

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="https://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>true</lpg:constrained>
       </per-node-capabilities>
       <per-node-capabilities>
         <node-selector>/es:audit-logs/es:audit-log/es:timestamp</node-\
   selector>
         <lpg:indexed>true</lpg:indexed>
       </per-node-capabilities>
       <per-node-capabilities>
         <node-selector>/es:audit-logs/es:audit-log/es:member-id</node-\
   selector>
         <lpg:indexed>true</lpg:indexed>
       </per-node-capabilities>
       <per-node-capabilities>
         <node-selector>/es:audit-logs/es:audit-log/es:outcome</node-se\
   lector>
         <lpg:indexed>true</lpg:indexed>
       </per-node-capabilities>
     </datastore-capabilities>
   </system-capabilities>
```

4.3. YANG Module

This YANG module has normative references to [RFC7952] and [RFC9196].

```
   <CODE BEGINS> file "ietf-list-pagination@2026-06-04.yang"

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

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

     import ietf-datastores {
       prefix ds;
       reference
```

```
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }

  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
      "RFC 9196: YANG Modules Describing Capabilities for Systems and
                 Datastore Update Notifications";
  }

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

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>

     Author:   Kent Watsen
               <kent+ietf@watsen.net>

     Author:   Qin Wu
               <bill.wu@huawei.com>

     Author:   Per Andersson
               <per.ietf@ionio.se>

     Author:   Olof Hagsand
               <olof@hagsand.se>

     Author:   Hongwei Li
               <flycoolman@gmail.com>";

  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) 2026 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

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

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

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

      revision 2026-06-04 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: List Pagination for YANG-driven Protocols";
      }

      // Features

      feature sort-by {
        description
          "This feature indicates that the parameters 'locale' and
           'sort-by' are supported.";
      }

      feature where {
        description
          "This feature indicates that the parameter 'where' is
           supported";
      }
      // Annotations

      md:annotation remaining {
        type union {
```

```
        type uint32;
        type enumeration {
          enum "unknown" {
            description
              "Indicates that the 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.";
      reference
        "RFC XXXX Section 3.1.7 The 'limit' Query Parameter";
    }

    md:annotation next {
      type string;
      description
        "This annotation contains the opaque encoded value of the next
         cursor in the pagination.";
      reference
        "RFC XXXX Section 3.1.6 The 'cursor' Query Parameter";
    }

    md:annotation previous {
      type string;
      description
        "This annotation contains the opaque encoded value of the
         previous cursor in the pagination.";
      reference
        "RFC XXXX Section 3.1.6 The 'cursor' Query Parameter";
    }

    md:annotation locale {
      if-feature "sort-by";
      type string;
      description
        "This annotation contains the locale used when sorting.
```

```
          The format is a free form string but SHOULD follow the
          language sub-tag format defined in RFC 5646.
          An example is 'sv_SE'.

          For further details see references:
          RFC 5646: Tags for identifying Languages
          RFC 6365: Technology Used in Internationalization in the
                    IETF";
        reference
          "RFC XXXX Section 3.1.3 The 'locale' Query Parameter";
      }

      // 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.";
      }

      identity cursor-not-found {
        base list-pagination-error;
        description
          "The 'cursor' query parameter value is unknown for the target
           list.";
      }

      identity locale-unavailable {
        if-feature "sort-by";
        base list-pagination-error;
        description
          "The 'locale' query parameter input is not a valid
           locale or the locale is not available on the system.";
      }

      // Groupings

      grouping pagination-parameters {
        description
          "This grouping may be used by protocol-specific YANG modules
           to define a protocol-specific pagination parameters.";
        container list-pagination {
```

```
        description
          "List pagination parameters.";
        leaf where {
          if-feature "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.

             If the XPath expression references a node identifier that
             does not exist in the schema, or is optional or
             conditional in the schema, the filter evaluates to false
             and nothing is filtered from the result-set.

             It is an error if the XPath expression references
             constrained 'config false' lists and leaf-lists, if the
             node identifier does not point to a node having the
             'indexed' leaf set to 'true' (see RFC XXXX).";
        }
        leaf locale {
          when "(../sort-by)" {
              description
              " The 'locale' parameter may only be specified when the
                'sort-by' parameter is specified.";
              }
          type string;
          description
            "The 'locale' parameter indicates the locale which the
             entries in the working result-set should be collated.";
        }
        leaf sort-by {
          if-feature "sort-by";
          type union {
            type string {
              // An RFC 7950 'descendant-schema-nodeid'.
              pattern '([A-Za-z_][A-Za-z0-9_.-]*:)?'
                    + '[A-Za-z_][A-Za-z0-9_.-]*'
                    + '(/[A-Za-z_]([A-Za-z0-9_.-]*:)?'
```

```
                  + '[A-Za-z_]([A-Za-z0-9_.-]*))*';
          }
          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).";
      }
      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.";
      }
       choice navigation-type {
        case cursor {
         leaf cursor {
           type string;
           default "";
           description
             "The 'cursor' parameter indicates where to start the
              working result-set (i.e. after the 'direction' paramet\
```

```
   er
                has been applied), the elements before the cursor are
                skipped over when preparing the response. Furthermore \
   the
                result-set is annotated with attributes for the next a\
   nd
                previous cursors following a result-set constrained wi\
   th
                the 'limit' query parameter.";
             }
           }
           case offset {
            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.";
             }
           }
           description
             "Indicates is navigation is by 'offset' or 'cursor'.  Whil\
   st
              both options have a default value indicating to start at \
   the
              beginning of the result set, the 'offset' option is defau\
   lt
              so that there is no expectation that the cursor's next/pr\
   ev
              values will be returned.";
          }
         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.";
        }
        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 descendant 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
      "/sysc:system-capabilities/sysc:datastore-capabilities"
      + "/sysc:per-node-capabilities" {

      // Ensure the following nodes are only used for the
      // <operational> datastore.
      when "/sysc:system-capabilities/sysc:datastore-capabilities"
           + "/sysc:datastore = 'ds:operational'";

      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 boolean;
        default false;
        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'.

           Setting this to true will make the list constrained.

           By default this is false, i.e. a list is not constrained.";
      }
      leaf indexed {
        type boolean;
        default false;
        description
          "Indicates that the targeted descendant node of a
           'constrained' list (see the 'constrained' leaf) may be
           used in 'where' filters and/or 'sort-by' parameters.
           If a descendant node of a 'constrained' list is not
           'indexed', then it MUST NOT be used in 'where' filters
           or 'sort-by' parameters.

           Setting this to true will allow a constrained list to be
           used in 'where' filters and/or 'sort-by' parameters.

           By default this is false, i.e. a list constrained is not
           allowed for 'where' filters or 'sort-by' parameters.";
      }
      leaf cursor-supported {
        type boolean;
        default false;
        description
          "Indicates that the targeted list node supports the
           'cursor' parameter.

           Setting this to true will enable the 'cursor' query
           parameter for a 'config false' list.

           By default this is false, i.e. the 'cursor' query parameter
           is disabled for 'config false' lists.";
      }
    }
  }

  <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. Considerations for the "ietf-list-pagination" YANG Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

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

All protocol-accessible data nodes in this module are read-only and cannot be modified. Access control may be configured to avoid exposing any read-only data that is defined by the augmenting module documentation as being security sensitive.

Since this module also defines groupings, these considerations are primarily for the designers of other modules that use these groupings.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

This module does not define any RPCs or actions or notifications, and thus the security consideration for such is not provided here.

7. References

7.1. 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>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/info/rfc5646>.

[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>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

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

[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>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", 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>.

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

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>.

[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>.

[REST-Dissertation] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>.

[I-D.ietf-netconf-list-pagination-nc] Watsen, K., Wu, Q., Andersson, P., Hagsand, O., and H. Li, "NETCONF Extensions to Support List Pagination", Work in Progress, Internet-Draft, draft-ietf-netconf-list-pagination-nc-11, 1 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-list-pagination-nc-11>.

[I-D.ietf-netconf-list-pagination-rc] Watsen, K., Wu, Q., Andersson, P., Hagsand, O., and H. Li, "RESTCONF Extensions to Support List Pagination", Work in Progress, Internet-Draft, draft-ietf-netconf-list-pagination-rc-10, 12 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-list-pagination-rc-10>.

[I-D.ietf-netconf-restconf-collection] Bierman, A., Bj旦rklund, 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>.

[I-D.zheng-netconf-fragmentation] Zheng, G. and Z. Wang, "A NETCONF Extension for Data Fragmentation", Work in Progress, Internet-Draft, draft-zheng-netconf-fragmentation-02, 26 June 2018, <https://datatracker.ietf.org/doc/html/draft-zheng-netconf-fragmentation-02>.

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 peculiarities 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 "https://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 visible 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 member 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 int64 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@users.example.net",
          "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"
          }
        },
        {
          "member-id": "奪sa",
          "email-address": "asa@users.example.net",
          "password": "$0$1543",
          "avatar": "BASE64VALUE=",
          "privacy-settings": {
            "post-visibility": "unlisted"
          },
          "following": ["alice", "bob"],
          "stats": {
            "joined": "2022-02-19T13:12:00Z",
            "membership-level": "standard",
            "last-activity": "2022-04-19T13:12:59Z"
          }
        }
      ]
    },
```

```
    "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.

The "members" list in the example data set is "ordered-by system" and the queries without explicit "sort-by" below return them in this given order. Note that the order of the result-set without explicit "sort-by" is implementation specific for "ordered-by system" lists.

> | Note that the user "奪sa" is only included in Appendix A.3.7.
> | This to isolate the parameter in the example query and its
> | behavior.

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.

> | 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-type: application
   error-tag: invalid-value
   error-app-tag: ietf-list-pagination:offset-out-of-range
```

A.3.3. The "cursor" Parameter

Noting that "cursor" is an opaque encoded value represented by a string, which addresses an element in a list.

> | The default value is empty, which is the same as supplying the
> | cursor value for the first element in the list.

These vector tests assume the target "/example-social:members/member" which has five members.

> | Note that response has added attributes describing the result
> | set and position in pagination.

A.3.3.1. Cursor using the Default Value

REQUEST

```
Target: /example-social:members/member
  Pagination Parameters:
    Where:     -
    Sort-by:   -
    Direction: -
    Offset:    -
    Limit:     2
    Cursor:    -
```

RESPONSE

```
{
  "example-social: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"
        }
      }
    ],
    "@example-social:member": [
      {
        "ietf-list-pagination:remaining": 3,
        "ietf-list-pagination:previous": "",
        "ietf-list-pagination:next": "YWxpY2U="
      }
    ]
  }
```

A.3.3.2. Cursor Using the "next" Value

```
REQUEST

Target: /example-social:members/member
  Pagination Parameters:
    Where:     -
    Sort-by:   -
    Direction: -
    Offset:    -
    Limit:     2
    Cursor:    YWxpY2U=
```

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", "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"
        }
      }
    ],
    "@example-social:member": [
      {
        "ietf-list-pagination:remaining": 1,
        "ietf-list-pagination:previous": "ZXJpYw==",
        "ietf-list-pagination:next": "am9l"
      }
    ]
  }
```

A.3.3.3. Cursor Using Another "next" Value

REQUEST

```
Target: /example-social:members/member
  Pagination Parameters:
    Where:     -
    Sort-by:   -
    Direction: -
    Offset:    -
    Limit:     2
    Cursor:    am9l
```

RESPONSE

```
{
  "example-social:member": [
    {
      "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:member": [
    {
      "ietf-list-pagination:remaining": 0,
      "ietf-list-pagination:previous": "bGlu",
      "ietf-list-pagination:next": ""
    }
  ]
}
```


A.3.3.4. Cursor Using an Invalid Value

REQUEST

The cursor used does not exist in the datastore.

```
Target: /example-social:members/member
  Pagination Parameters:
    Where:     -
    Sort-by:   -
    Direction: -
    Offset:    -
    Limit:     -
    Cursor:    <invalid-value>

RESPONSE

error-type: application
error-tag: invalid-value
error-app-tag: ietf-list-pagination:cursor-not-found
```

A.3.4. 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.3.4.1. 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.3.4.2. 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.3.5. 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.3.5.1. 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 descendant and the other using an indirect descendant.

A.3.5.1.1. 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.3.5.1.2. type is a "list" and sort-by node is a direct descendant

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

This vector test uses the target "/example-social:members/member", which is a "list", and the sort-by descendant 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.3.5.1.3. type is a "list" and sort-by node is an indirect descendant

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

This vector test uses the target "/example-social:members/member", which is a "list", and the sort-by descendant node "stats/joined", which is a "config false" descendant 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.3.6. 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.3.6.1. match of leaf-list’s values

This example selects the uint8-numbers greater than 7 in the member alice’s favorites.

REQUEST

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

RESPONSE

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

A.3.6.2. match on descendant 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",
      ...
    }
  ]
}
```

A.3.6.3. 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.3.7. The "locale" Parameter

The "locale" parameter may be used on any target node.

If this parameter is omitted, there is no default value and it is up to the server to choose a locale. This locale is then reported in the result-set as the "locale" metadata value.

Note that for ordered-by user lists and leaf-lists "locale" is not relevant, since the order is set by the user. For ordered-by system lists and leaf-lists, the server MAY report "locale" if the order that the server has chosen follows a valid locale,

If "locale" is used on an ordered-by user list, error-type "application" and error-tag "invalid-value" is returned.

If an ordered-by system target is not ordered according to any locale, the server omits the locale from the response.

REQUEST

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

RESPONSE

```
{
  "example-social:member": [
    {
      "member-id": "alice",
      ...
    },
    {
      "member-id": "bob",
      ...
    },
    {
      "member-id": "eric",
      ...
    },
    {
      "member-id": "joe",
      ...
    },
    {
      "member-id": "lin",
      ...
    },
    {
      "member-id": "奪sa",
      ...
    }
  ],
  "@example-social:member": [
    {
      "ietf-list-pagination:locale": "sv_SE"
    }
  ]
}
```

REQUEST

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

RESPONSE

```
{
  "example-social:member": [
    {
      "member-id": "alice",
      ...
    },
    {
      "member-id": "奪sa",
      ...
    },
    {
      "member-id": "bob",
      ...
    },
    {
      "member-id": "eric",
      ...
    },
    {
      "member-id": "joe",
      ...
    },
    {
      "member-id": "lin",
      ...
    }
  ],
  "@example-social:member": [
    {
      "ietf-list-pagination:locale": "en_US"
    }
  ]
}
```

REQUEST

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

RESPONSE

```
   error-type: application
   error-tag: invalid-value
   error-app-tag: ietf-list-pagination:locale-unavailable

   REQUEST

   This example targets an "ordered-by user" list.

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

   RESPONSE

   error-type: application
   error-tag: invalid-value

   REQUEST

   This example supplies "locale" but not "sort-by".

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

   RESPONSE

   error-type: application
   error-tag: invalid-value
```

A.3.8. The "sublist-limit" Parameter

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

A.3.8.1. 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 descendant lists and leaf-lists.

Note that, in the response, the "remaining" metadata value is set on the first element of each descendant 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.3.8.2. target is a datastore

This example uses the target node <intended> datastore.

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

Note that, in the response, the "remaining" metadata value is set on the first element of each descendant 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.3.9. Combinations of Parameters

A.3.9.1. 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

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 (ordered by first name): Andy Bierman, Martin Bj旦rklund, Robert Varga, Rob Wills, and Rob Wilton. The authors would also like to thank Jen Linkova for the OPS Directorate review.

Authors’ Addresses

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net

Qin Wu
Huawei Technologies
Email: bill.wu@huawei.com

Per Andersson
Ionio Systems
Email: per.ietf@ionio.se

Olof Hagsand
SUNET
Email: olof@hagsand.se

Hongwei Li
Hewlett Packard Enterprise
Email: flycoolman@gmail.com