Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 22, 2021

A. Clemm
Y. Qu
Futurewei
J. Tantsura
Apstra
A. Bierman
YumaWorks

September 18, 2020

Comparison of NMDA datastores
draft-ietf-netmod-nmda-diff-06

Abstract

This document defines an RPC operation to compare management datastores that comply with the NMDA architecture.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 22, 2021.

Copyright Notice

Table of Contents

1. Introduction 2
2. Key Words 3
3. Definitions and Acronyms 3
4. Data Model Overview 4
5. YANG Data Model 6
6. Example 10
7. Performance Considerations 14
8. Possible Future Extensions 15
9. IANA Considerations 15
9.1. Updates to the IETF XML Registry 15
9.2. Updates to the YANG Module Names Registry 15
10. Security Considerations 16
11. Acknowledgments 16
12. References 16
12.1. Normative References 17
12.2. Informative References 18
Authors' Addresses 18

1. Introduction

The revised Network Management Datastore Architecture (NMDA) [RFC8342] introduces a set of new datastores that each hold YANG-defined data [RFC7950] and represent a different "viewpoint" on the data that is maintained by a server. New YANG datastores that are introduced include <intended>, which contains validated configuration data that a client application intends to be in effect, and <operational>, which contains at least conceptually operational state data (such as statistics) as well as configuration data that is actually in effect.

NMDA introduces in effect a concept of "lifecycle" for management data, allowing to clearly distinguish between data that is part of a configuration that was supplied by a user, configuration data that has actually been successfully applied and that is part of the operational state, and overall operational state that includes both applied configuration data as well as status and statistics.

As a result, data from the same management model can be reflected in multiple datastores. Clients need to specify the target datastore to be specific about which viewpoint of the data they want to access. This way, an application can differentiate whether they are (for example) interested in the configuration that has been applied and is

actually in effect, or in the configuration that was supplied by a client and that is supposed to be in effect.

Due to the fact that data can propagate from one datastore to another, it is possibly for differences between datastores to occur. Some of this is entirely expected, as there may be a time lag between when a configuration is given to the device and reflected in <intended>, until when it actually takes effect and is reflected in <operational>. However, there may be cases when a configuration item that was to be applied may not actually take effect at all or needs an unusually long time to do so. This can be the case due to certain conditions not being met, resource dependencies not being resolved, or even implementation errors in corner conditions.

When configuration that is in effect is different from configuration that was applied, many issues can result. It becomes more difficult to operate the network properly due to limited visibility of actual status which makes it more difficult to analyze and understand what is going on in the network. Services may be negatively affected (for example, breaking a service instance resulting in service is not properly delivered to a customer) and network resources be misallocated.

Applications can potentially analyze any differences between two datastores by retrieving the contents from both datastores and comparing them. However, in many cases this will be at the same time costly and extremely wasteful.

This document introduces a YANG data model which defines RPCs, intended to be used in conjunction with NETCONF [RFC6241] or RESTCONF [RFC8040], that allow a client to request a server to compare two NMDA datastores and report any differences.

2. Key Words

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. Definitions and Acronyms

NMDA: Network Management Datastore Architecture

RPC: Remote Procedure Call

4. Data Model Overview

At the core of the solution is a new management operation, <compare>, that allows to compare two datastores for the same data. The operation checks whether there are any differences in values or in data nodes that are contained in either datastore, and returns any differences as output. The output is returned in the format specified in YANG-Patch [RFC8072].

The YANG data model defines the <compare> operation as a new RPC. The operation takes the following input parameters:

- source: The source identifies the datastore that will serve as reference for the comparison, for example <intended>.

- target: The target identifies the datastore to compare against the source.

- filter-spec: This is a choice between different filter constructs to identify the portions of the datastore to be retrieved. It acts as a node selector that specifies which data nodes are within the scope of the comparison and which nodes are outside the scope. This allows a comparison operation to be applied only to a specific portion of the datastore that is of interest, such as a particular subtree. (The filter dow not contain expressions that would match values data nodes, as this is not required by most use cases and would complicate the scheme, from implementation to dealing with race conditions.)

- all: When set, this parameter indicates that all differences should be included, including differences pertaining to schema nodes that exist in only one of the datastores. When this parameter is not included, a prefiltering step is automatically applied to exclude data from the comparison that does not pertain to both datastores: if the same schema node is not present in both datastores, then all instances of that schema node and all its descendants are excluded from the comparison. This allows client applications to focus on the differences that constitute true mismatches of instance data without needing to specify more complex filter constructs.

- exclude-origin: When set, this parameter indicates that origin metadata should not not be included as part of RPC output. When this parameter is omitted, origin metadata in comparisons that involve <operational> is by default included.

The operation provides the following output parameter:

- differences: This parameter contains the list of differences. Those differences are encoded per YANG-Patch data model defined in RFC8072. When a datastore node in the source of the comparison is not present in the target of the comparison, this can be indicated either as a "delete" or as a "remove" in the patch as there is no differentiation between those operations for the purposes of the comparison. The YANG-Patch data model is augmented to indicate the value of source datastore nodes in addition to the patch itself that would need to be applied to the source to produce the target. When the target datastore is <operational>, "origin" metadata is included as part of the patch. Including origin metadata can help in some cases explain the cause of a difference, for example when a data node is part of <intended> but the origin of the same data node in <operational> is reported as "system".

The data model is defined in the ietf-nmda-compare YANG module. Its structure is shown in the following figure. The notation syntax follows [RFC8340].

```
module: ietf-nmda-compare
rpcs:
   +---x compare
      +---w input
      |  +---w source            identityref
      |  +---w target            identityref
      |  +---w all?              empty
      |  +---w exclude-origin?   empty
      |  +---w (filter-spec)?
      |     +--:(subtree-filter)
      |     |  +---w subtree-filter?
      |     +--:(xpath-filter)
      |        +---w xpath-filter?     yang:xpath1.0 {nc:xpath}?
      +--ro output
         +--ro (compare-response)?
            +--:(no-matches)
            |  +--ro no-matches?    empty
            +--:(differences)
               +--ro differences
                  +--ro yang-patch
                     +--ro patch-id    string
                     +--ro comment?    string
                     +--ro edit* [edit-id]
                        +--ro edit-id         string
                        +--ro operation       enumeration
                        +--ro target          target-resource-offset
                        +--ro point?          target-resource-offset
                        +--ro where?          enumeration
                        +--ro value?
                        +--ro source-value?
```

Structure of ietf-nmda-compare

5. YANG Data Model

```
<CODE BEGINS> file "ietf-nmda-compare@2020-09-18.yang"
module ietf-nmda-compare {

  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-nmda-compare";

  prefix cmp;

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

```
      prefix ds;
      reference "RFC 8342: Network Management Datastore
                 Architecture (NMDA)";
    }
    import ietf-yang-patch {
      prefix ypatch;
      reference "RFC 8072: YANG Patch Media Type";
    }
    import ietf-netconf {
      prefix nc;
      reference "RFC6241: Network Configuration Protocol (NETCONF)";
    }

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

       Author: Alexander Clemm
               <mailto:ludwig@clemm.org>

       Author: Yingzhen Qu
               <mailto:yqu@futurewei.com>

       Author: Jeff Tantsura
               <mailto:jefftant.ietf@gmail.com>

       Author: Andy Bierman
               <mailto:andy@yumaworks.com>";

    description
      "The YANG data model defines a new operation, <compare>, that
       can be used to compare NMDA datastores.

       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.

       Copyright (c) 2020 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 2020-09-18 {
        description
          "Initial revision";
        reference
          "RFC XXXX: Comparison of NMDA datastores";
      }

      /* RPC */
      rpc compare {
        description
          "NMDA compare operation.";
        input {
          leaf source {
            type identityref {
              base ds:datastore;
            }
            mandatory true;
            description
              "The source datastore to be compared.";
          }
          leaf target {
            type identityref {
              base ds:datastore;
            }
            mandatory true;
            description
              "The target datastore to be compared.";
          }
          leaf all {
            type empty;
            description
              "When this leaf is provided, all data nodes are compared,
               whether their schema node pertains to both datastores or
               not. When this leaf is omitted, a prefiltering step is
               automatically applied that excludes data nodes from the
               comparison that can occur in only one datastore but not
               the other.  Specifically, if one of the datastores
               (source or target) contains only configuration data and
               the other datastore is <operational>, data nodes for
               which config is false are excluded from the comparison.";
          }
          leaf exclude-origin {
            type empty;
```

```
        description
          "When this leaf is provided, origin metadata is not
           included as part of RPC output. When this leaf is
           omitted, origin metadata in comparisons that involve
           <operational> is by default included.";
      }
      choice filter-spec {
        description
          "Identifies the portions of the datastores to be
               compared.";
        anydata subtree-filter {
          description
            "This parameter identifies the portions of the
             target datastore to retrieve.";
          reference "RFC 6241, Section 6.";
        }
        leaf xpath-filter {
          if-feature nc:xpath;
          type yang:xpath1.0;
          description
            "This parameter contains an XPath expression
             identifying the portions of the target
             datastore to retrieve.";
          reference "RFC 6021: Common YANG Data Types";
        }
      }
    }
    output {
      choice compare-response {
        description
          "Comparison results.";
        leaf no-matches {
          type empty;
          description
            "This leaf indicates that the filter did not match
             anything and nothing was compared.";
        }
        container differences {
          description
            "The list of differences, encoded per RFC8072 with an
             augmentation to include source values where applicable.
             When a datastore node in the source is not present in
             the target, this can be indicated either as a 'delete'
             or as a 'remove' as there is no difference between
             them for the purposes of the comparison.";
          uses ypatch:yang-patch {
            augment "yang-patch/edit" {
              description
```

```
                  "Provide the value of the source of the patch,
                   respectively of the comparison, in addition to
                   the target value, where applicable.";
                anydata source-value {
                  when "../operation = 'delete'"
                    + "or ../operation = 'merge'"
                    + "or ../operation = 'move'"
                    + "or ../operation = 'replace'"
                    + "or ../operation = 'remove'";
                  description
                    "The anydata 'value' is only used for 'delete',
                     'move', 'merge', 'replace', and 'remove'
                     operations.";
                }
                reference "RFC 8072: YANG Patch Media Type";
              }
            }
          }
        }
      }
    }
  }
<CODE ENDS>
```

6. Example

The following example compares the difference between <operational> and <intended> for a subtree under "interfaces". The subtree contains a subset of objects that are defined in a YANG data model for the management of interfaces defined in [RFC8343]. The excerpt of the data model whose instantiation is basis of the comparison is as follows:

```
   container interfaces {
     description
       "Interface parameters.";
     list interface {
       key "name";
       leaf name {
         type string;
         description
           "The name of the interface".
       }
       leaf description {
         type string;
         description
           "A textual description of the interface.";
       }
       leaf enabled {
         type boolean;
         default "true";
         description
           "This leaf contains the configured, desired state of the
            interface.";"
       }
     }
   }
```

The contents of <intended> and <operational> datastores:

```
   //INTENDED
   <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
     <interface>
       <name>eth0</name>
       <enabled>false</enabled>
       <description>ip interface</description>
     </interface>
   </interfaces>

   //OPERATIONAL
   <interfaces
       xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
       xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">
     <interface or:origin="or:learned">
       <name>eth0</name>
       <enabled>true</enabled>

       </interface>
   </interfaces>
```

<operational> does not contain object "description" that is contained in <intended>. Another object, "enabled", has differences in values, being "true" in <operational> and "false" in <intended>. A third object, "name", is the same in both cases. The origin of the objects in <operational> is "learned", which may help explain the discrepancies.

RPC request to compare <operational> (source of the comparison) with <intended>(target of the comparison):

```
<rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <compare xmlns="urn:ietf:params:xml:ns:yang:ietf-nmda-compare"
      xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
    <source>ds:operational</source>
    <target>ds:intended</target>
    <xpath-filter
        xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      /if:interfaces
    </xpath-filter>
  </compare>
</rpc>
```

RPC reply, when a difference is detected:

```
   <rpc-reply
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
        message-id="101">
     <differences
        xmlns="urn:ietf:params:xml:ns:yang:ietf-nmda-compare"
        xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">
       <yang-patch>
         <patch-id>interface status</patch-id>
         <comment>
           diff between operational (source) and intended (target)
         </comment>
         <edit>
           <edit-id>1</edit-id>
           <operation>replace</operation>
           <target>/ietf-interfaces:interface=eth0/enabled</target>
           <value>
             <if:enabled>false<if:enabled>
           </value>
           <source-value>
              <if:enabled or:origin="or:learned">true</if:enabled>
           </source-value>
           <edit-id>2</edit-id>
           <operation>create</operation>
           <target>/ietf-interfaces:interface=eth0/description</target>
           <value>
             <if:description>ip interface<description>
           </value>
         </edit>
       </yang-patch>
     </differences>
   </rpc-reply>

   The same request in RESTCONF (using JSON format):

   POST /restconf/operations/ietf-nmda-compare:compare HTTP/1.1
   Host: example.com
   Content-Type: application/yang-data+json
   Accept: application/yang-d
   { "ietf-nmda-compare:input" {
      "source" : "ietf-datastores:operational"
      "target" : "ietf-datastores:intended"
      "xpath-filter" : \
          "/ietf-interfaces:interfaces"
      }
   }

   The same response in RESTCONF (using JSON format):
```

```
  HTTP/1.1 200 OK
  Date: Thu, 26 Jan 2019 20:56:30 GMT
  Server: example-server
  Content-Type: application/yang-d
  { "ietf-nmda-compare:output" : {
      "differences" : {
        "ietf-yang-patch:yang-patch" : {
          "patch-id" : "interface status",
          "comment" : "diff between intended (source) and operational",
          "edit" : [
            {
              "edit-id" : "1",
              "operation" : "replace",
              "target" : "/ietf-interfaces:interface=eth0/enabled",
              "value" : {
                 "ietf-interfaces:interface/enabled" : "false"
              },
              "source-value" : {
                 "ietf-interfaces:interface/enabled" : "true",
                 "@ietf-interfaces:interface/enabled" : {
                   "ietf-origin:origin" : "ietf-origin:learned"
                 }
               }
              "edit-id" : "2",
              "operation" : "create",
              "target" : "/ietf-interfaces:interface=eth0/description",
              "value" : {
                 "ietf-interface:interface/description" : "ip interface"
              }
            }
          ]
        }
      }
    }
  }
```

7. Performance Considerations

The compare operation can be computationally expensive. While responsible client applications are expected to use the operation responsibly and sparingly only when warranted, implementations need to be aware of the fact that excessive invocation of this operation will burden system resources and need to ensure that system performance will not be adversely impacted. One possibility for an implementation to mitigate against such a possibility is to limit the number of requests that is served to a client, or to any number of clients, in any one time interval, rejecting requests made at a higher frequency than the implementation can reasonably sustain.

8. Possible Future Extensions

It is conceivable to extend the compare operation with a number of possible additional features in the future.

Specifically, it is possible to define an extension with an optional feature for dampening. This will allow clients to specify a minimum time period for which a difference must persist for it to be reported. This will enable clients to distinguish between differences that are only fleeting from ones that are not and that may represent a real operational issue and inconsistency within the device.

For this purpose, an additional input parameter can be added to specify the dampening period. Only differences that pertain for at least the dampening time are reported. A value of 0 or omission of the parameter indicates no dampening. Reporting of differences MAY correspondingly be delayed by the dampening period from the time the request is received.

To implement this feature, a server implementation might run a comparison when the RPC is first invoked and temporarily store the result. Subsequently, it could wait until after the end of the dampening period to check whether the same differences are still observed. The differences that still persist are then returned.

9. IANA Considerations

9.1. Updates to the IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-nmda-compare

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

9.2. Updates to the YANG Module Names Registry

This document registers a YANG module in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the following registration is requested:

name: ietf-nmda-compare

namespace: urn:ietf:params:xml:ns:yang:ietf-nmda-compare

prefix: cmp

reference: RFC XXXX

10. 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 access control model [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.

The RPC operation defined in this YANG module, "compare", may be considered sensitive or vulnerable in some network environments. It is thus important to control access to this operation. This is the sensitivity/vulnerability of RPC operation "compare":

Comparing datastores for differences requires a certain amount of processing resources at the server. An attacker could attempt to attack a server by making a high volume of comparison requests. Server implementations can guard against such scenarios in several ways. For one, they can implement the NETCONF access control model in order to require proper authorization for requests to be made. Second, server implementations can limit the number of requests that they serve to a client in any one time interval, rejecting requests made at a higher frequency than the implementation can reasonably sustain.

11. Acknowledgments

We thank Rob Wilton, Martin Bjorklund, Mahesh Jethanandani, Lou Berger, Kent Watsen, Phil Shafer, Ladislav Lhotka, Tim Carey, and Reshad Rahman for valuable feedback and suggestions.

12. References

12.1. 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>.

[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>.

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

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

[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>.

[RFC8072] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", RFC 8072, DOI 10.17487/RFC8072, February 2017, <https://www.rfc-editor.org/info/rfc8072>.

[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>.

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

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

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

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

12.2. Informative References

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

Authors' Addresses

Alexander Clemm
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: ludwig@clemm.org

Yingzhen Qu
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
USA

Email: yqu@futurewei.com

Jeff Tantsura
Apstra

Email: jefftant.ietf@gmail.com

Andy Bierman
YumaWorks

Email: andy@yumaworks.com