Internet Engineering Task Force (IETF)
Request for Comments: 9144
Category: Standards Track
ISSN: 2070-1721

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

December 2021

Comparison of Network Management Datastore Architecture (NMDA) Datastores

Abstract

This document defines a Remote Procedure Call (RPC) operation to compare management datastores that comply with the Network Management Datastore Architecture (NMDA).

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9144.

Table of Contents

1. Introduction

The revised 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 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, distinguishing 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 the overall operational state that includes 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. For example, a client application can differentiate whether they are interested in the configuration that is supplied to a server and is supposed to be in effect or the configuration that has been applied and is actually in effect on the server.

Due to the fact that data can propagate from one datastore to another, it is possible 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, certain parts of the configuration not propagating because they are considered inactive, resource dependencies not being resolved, or even implementation errors in corner conditions.

When the configuration that is in effect is different from the configuration that was applied, many issues can result. It becomes more difficult to operate the network properly due to limited visibility of the actual operational status, which makes it more difficult to analyze and understand what is going on in the network. Services may be negatively affected (for example, degrading or

breaking a customer service), and network resources may 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 both costly and extremely wasteful.

This document introduces a YANG data model that defines RPCs intended to be used in conjunction with NETCONF [RFC6241] or RESTCONF [RFC8040]. These RPCs 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. Data Model Overview

The core of the solution is a new management operation, <compare>, that compares the data tree contents of two datastores. 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 to serve as the reference for the comparison -- for example, <intended>.

target: The target identifies the datastore to compare against the source -- for example, <operational>.

filter-spec: This is a choice between different filter constructs to identify the parts 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 part of the datastore that is of interest, such as a particular subtree. Note that the filter does not allow expressions that match against data node values, since this may incur implementation difficulties and is not required for normal use cases.

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.

report-origin: When set, this parameter indicates that origin metadata should be included as part of RPC output. When this parameter is omitted, origin metadata in comparisons that involve <operational> is by default omitted. Note that origin metadata only applies to <operational>; it is therefore also omitted in comparisons that do not involve <operational> regardless of whether or not the parameter is set.

The operation provides the following output parameter:

differences: This parameter contains the list of differences. Those differences are encoded per the 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> and the input parameter "report-origin" is set, origin metadata is included as part of the patch. Including origin metadata can help explain the cause of a difference in some cases -- 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 report-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?
```

Figure 1: Structure of ietf-nmda-compare

4. YANG Data Model

This YANG module includes references to [RFC6991], [RFC8342], [RFC8072], and [RFC6241].

```
<CODE BEGINS> file "ietf-nmda-compare@2021-12-10.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
      "RFC 6241: Network Configuration Protocol (NETCONF)";
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@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.

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

   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject to
   the license terms contained in, the 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 9144; see the
   RFC itself for full legal notices.";

revision 2021-12-10 {
  description
    "Initial revision.";
  reference
    "RFC 9144: Comparison of Network Management Datastore
     Architecture (NMDA) Datastores";
}

/* RPC */

rpc compare {
  description
    "NMDA datastore 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
         the config that is false are excluded from the
         comparison.";
    }
    leaf report-origin {
      type empty;
      description
        "When this leaf is provided, origin metadata is
         included as part of RPC output.  When this leaf is
         omitted, origin metadata in comparisons that involve
         <operational> is by default omitted.";
    }
    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 6991: 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 RFC 8072 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
                "Provides the value of the source of the patch,
                 respectively of the source 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>
```

5. 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]. For the purposes of understanding the subsequent example, the following excerpt of the data model whose instantiation is the basis of the comparison is provided:

```
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 in XML [W3C.REC-xml-20081126]:

```
<!--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 an instance for leaf "description" that is contained in <intended>. Another leaf, "enabled", has different values in the two datastores, being "true" in <operational> and "false" in <intended>. A third leaf, "name", is the same in both cases. The origin of the leaf instances 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>
    <report-origin/>
    <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>
      <edit>
        <edit-id>2</edit-id>
        <operation>create</operation>
        <target>/ietf-interfaces:interface=eth0/description</target>
        <value>
          <if:description>ip interface</if:description>
        </value>
      </edit>
    </yang-patch>
  </differences>
</rpc-reply>
```

The same request in RESTCONF (using JSON format [RFC7951]):

```
POST /restconf/operations/ietf-nmda-compare:compare HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json
Accept: application/yang-data+json

{ "ietf-nmda-compare:input" : {
   "source" : "ietf-datastores:operational",
   "target" : "ietf-datastores:intended",
   "report-origin" : null,
   "xpath-filter" : "/ietf-interfaces:interfaces"
   }
}
```

The same response in RESTCONF (using JSON format):

```
HTTP/1.1 200 OK
Date: Thu, 24 Jan 2019 20:56:30 GMT
Server: example-server
Content-Type: application/yang-data+json

{ "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"
            }
          }
        ]
      }
    }
  }
}
```

6. 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 this is to limit the number of requests that are served to a client, or to any number of clients, in any one time interval, by rejecting requests made at a higher frequency than the implementation can reasonably sustain.

While useful, tools such as YANG data models that allow for the monitoring of server resources, system performance, and statistics about RPCs and RPC rates are outside the scope of this document. When defined, any such model should be general in nature and not

limited to the RPC operation defined in this document.

7. IANA Considerations

7.1. Update to the IETF XML Registry

IANA has registered the following URI in the "IETF XML Registry" [RFC3688]:

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

7.2. Update to the YANG Module Names Registry

IANA has registered the following YANG module in the "YANG Module Names" registry [RFC6020]:

name: ietf-nmda-compare
namespace: urn:ietf:params:xml:ns:yang:ietf-nmda-compare
prefix: cmp
reference: RFC 9144

8. 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 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.

NACM specifies access for the server in its entirety, and the same access rules apply to all datastores. Any subtrees to which a requestor does not have read access are silently skipped and not included in the comparison.

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 NACM 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.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

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

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

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

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

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

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

[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>.

[W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <https://www.w3.org/TR/2008/REC-xml-20081126>.

9.2. 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>.

Appendix A. 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.

Acknowledgments

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

Authors' Addresses

Alexander Clemm
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
United States of America

Email: ludwig@clemm.org

Yingzhen Qu
Futurewei
2330 Central Expressway
Santa Clara, CA 95050
United States of America

Email: yqu@futurewei.com

Jeff Tantsura
Microsoft

Email: jefftant.ietf@gmail.com

Andy Bierman
YumaWorks

Email: andy@yumaworks.com