Workgroup: NETMOD
Internet-Draft:
draft-ma-netmod-immutable-flag-00
Published: 10 February 2022
Intended Status: Standards Track
Expires: 14 August 2022
Authors: Q. Ma, Ed. Q. Wu H. Li
Huawei Huawei HPE

Immutable Metadata Annotation

Abstract

This document defines a metadata annotation [RFC7952] named "immutable" to indicate the immutability of a particular instantiated data node. Any instantiated data node annotated with immutable="true" by the server is read-only to the clients of YANG-driven management protocols, such as NETCONF, RESTCONF and other management operations (e.g., SNMP and CLI requests).

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 14 August 2022.

Copyright Notice

Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

YANG [RFC7950] is a data modeling language used to model both state and configuration data, based on the "config" statement. However, there are some configuration data which may not be writable to clients, e.g., the interface name and type values created by the system due to the hardware currently present in the device cannot be modified by clients, while configurations such as MTU created by the system are free to be modified by the client.

Allowing some configuration modifiable while others not is inconsistent and introduces ambiguity to clients.

To address this issue, this document defines a metadata annotation [RFC7952] named "immutable" to indicate the immutability characteristic of a particular instantiated data node. Any instantiated data node marked with immutable="true" by the server is read-only to the clients of YANG-driven management protocols, such as NETCONF, RESTCONF and other management operations (e.g., SNMP and CLI requests).

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 [RFC6241] and [RFC8341] and are not redefined here:

- configuration data
- access operation

The following terms are defined in this document:

immutable: A metadata annotation indicating the immutability of a data node. An immutable data node is read-only to clients. Note that "immutable" is used to annotate instances of YANG data nodes rather than schema nodes. For instance, a "list" data node may exist in multiple instances in the data tree, "immutable" can annotate some of the instances as read-only, while others are not.

2. "Immutable" Metadata Annotation Definition

The "immutable" flag is used to indicate the immutability of a particular instantiated data node. It applies to the container, anydata, anyxml, leaf, list and leaf-list entries. The values are boolean types indicating whether the data node instance is immutable or not.

When the client retrieves a particular datastore, immutable data node instances MUST be annotated with immutable="true" by the server. If the "immutable" metadata annotation is not specified, the default is the same as the "immutable" value of the parent node. The default "immutable" value for a top level data node is false.

Any data node instance annotated with "immutable=true" is read-only to clients, which means that the following access operations for a particular instance are not allowed:

- Create: add descendant node instances for a container instance annotated with "immutable";
- Delete: delete a data node instance annotated with "immutable" or its child node instances from a datastore, e.g., delete an immutable list or leaf-list entry;
- Update: update an existing data node instance annotated with "immutable" or its child node instances in a datastore, e.g., modify the value of an immutable leaf data node;

However the following operations SHOULD be allowed:

* Create an immutable data node with a same value initially set by the system if it doesn't exist in the datastore, e.g., explicitly configure a system generated interface name in <running>;

* Delete the parent node of an immutable data node unless the parent node is also annotated with "immutable=true", e.g., /interfaces/interface/type leaf instance is immutable, but the deletion to the /interfaces/interface list entry is allowed;

Comment: Should we allow the client delete an "immutable" system instantiated node in <running> (see Appendix A.1.3)?

Comment: Annotations can only be attached to individual list/leaf-list entry, how would the client know if it's to apply to the whole list/leaf-list, the list/leaf-list entries, or both?

Servers MUST reject any attempt to the "create", "delete" and "update" access operations on an immutable data node. The error reporting is performed at various different time according to the selected target datastore. If the target datastore is "running" or "startup", the server should reply with an "invalid-value" at a <edit-config> operation time. If the target datastore is "candidate", the "invalid-value" error response to update an immutable data node is delayed until a <commit> or <validate> operation takes place. For an example of an "invalid-value" error response, see Appendix A.1.2.

3. YANG Module

```
<CODE BEGINS> file="ietf-immutable@2022-01-05.yang"
  module ietf-immutable {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-immutable";
    prefix im;

    import ietf-yang-metadata {
      prefix md;
    }

    organization
      "IETF Network Modeling (NETMOD) Working Group";

    contact
      "WG Web: <https://datatracker.ietf.org/wg/netmod/>

       WG List: <mailto:netmod@ietf.org>

       Author: Qiufang Ma
               <mailto:maqiufang1@huawei.com>

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

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

    description
      "This module defines a metadata annotation named 'immutable'
       to indicate the immutability of a particular instantiated
       data node. Any instantiated data node marked with
       immutable='true' by the server is read-only to the clients
       of YANG-driven management protocols, such as NETCONF,
       RESTCONF as well as SNMP and CLI requests.

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

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

       This version of this YANG module is part of RFC HHHH
       (https://www.rfc-editor.org/info/rfcHHHH); 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 2022-01-05 {
      description
        "Initial revision.";
      reference
        "RFC XXX: Immutable Metadata Annotation";
    }

    md:annotation immutable {
      type boolean;
      description
        "The 'immutable' annotation indicates the immutability of an
         instantiated data node. Any data node instance marked as
         'immutable=true' is read-only to clients and cannot be
         created/deleted/changed through NETCONF, RESTCONF or CLI.
         It applies to the container, anydata, anyxml and leaf
         instances, list and leaf-list entries.
         If not specified for a given configuration data node
         instance, then the immutability is the same as its parent
         node instance in the data tree. The default for any top-level
         configuration data node instance is false if not specified.";
    }
  }

<CODE ENDS>
```

4. IANA Considerations

4.1. The "IETF XML" Registry

This document registers one XML namespace URN in the 'IETF XML registry', following the format defined in [RFC3688].

URI: urn:ietf:params:xml:ns:yang:ietf-immutable
Registrant Contact: The IESG.
XML: N/A, the requested URIs are XML namespaces.

4.2. The "YANG Module Names" Registry

This document registers one module name in the 'YANG Module Names' registry, defined in [RFC6020].

```
name: ietf-immutable
prefix: im
namespace: urn:ietf:params:xml:ns:yang:ietf-immutable
RFC: XXXX // RFC Ed.: replace XXXX and remove this comment
```

5. Security Considerations

The YANG module specified in this document defines a metadata annotation for data nodes that is designed to be accessed 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].

Since immutable information is tied to applied configuration values, it is only accessible to clients that have the permissions to read the applied configuration values.

The security considerations for the Defining and Using Metadata with YANG (see Section 9 of [RFC7952]) apply to the metadata annotation defined in this document.

6. References

6.1. 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>.

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

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

[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>.

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

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

6.2. Informative References

[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>.

Appendix A. Usage Examples

A.1. Interface Example

In this section, the following fragment of a fictional interface module is used:

```
    container interfaces {
      list interface {
        key "name";
        leaf name {
          type string;
        }
        leaf type {
          type identityref {
            base ianaift:iana-interface-type;
          }
        }
        leaf mtu {
          type uint16;
        }
        leaf-list ip-address {
          type inet:ip-address;
        }
      }
    }
```

In this example model, when an interface is physically present, the system will create an interface entry automatically with valid name and type values. Let's say that there is a system-defined interface "eth0" in <operational>:

```
<interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
        xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
        xmlns:im="urn:ietf:params:xml:ns:yang:ietf-immutable"
        or:origin="or:system">
  <interface>
    <name>eth0</name>
    <type im:immutable="true">ianaift:ethernetCsmacd</type>
    <mtu>1500</mtu>
  </interface>
</interfaces>
```

The "type" leaf instance is annotated with immutable="true" in the RPC response, which means that it is not allowed to be modified by client configuration operations. Note that although the "name" defined as a list key leaf isn't annotated with immutable="true", modification of the key value for a particular list entry is never allowed. Deletion to the whole list entry of interface "eth0" should be allowed in this case, since there is no annotation for the "type" leaf instance's parent node.

A.1.1. Creating an "immutable" Interface Type

The server should accept the configuration to set the leaf "type" to the value same as in <operational>:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     message-id="101">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interface xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"
            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
            xc:operation="create">
        <name>eth0</name>
        <type>ianaift:ethernetCsmacd</type>
      </interface>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

A.1.2. Changing an "immutable" Interface Type

If a client tries to change the type of an interface to a value that doesn't match the real type of the interface used by the system, the server must reject the request:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interface xc:operation="merge"
            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
        <name>eth0</name>
        <type>ianaift:tunnel</type>
      </interface>
    </config>
  </edit-config>
</rpc>

<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>invalid-value</error-tag>
    <error-severity>error</error-severity>
    <error-path xmlns:t="http://example.com/schema/1.2/config">
      /interfaces/interface[name="eth0"]/type
    </error-path>
    <error-message xml:lang="en">
      Invalid type for interface eth0
    </error-message>
  </rpc-error>
</rpc-reply>
```

A.1.3. Deleting an "immutable" Interface Type

Should the server accept the configuration of deleting the type of interface named "eth0" from the running configuration?

```
<rpc message-id="102"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <interface>
        <name>eth0</name>
        <type xc:operation="delete">ianaift:ethernetCsmacd</type>
      </interface>
    </config>
  </edit-config>
</rpc>
```

Even the type of interface named "eth0" is deleted from the running configuration, the client which performs a <get-data> towards <operational> will still get:

```
<interfaces xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
            xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
            xmlns:im="urn:ietf:params:xml:ns:yang:ietf-immutable"
            or:origin="or:intended">
  <interface>
    <name>eth0</name>
    <type im:immutable="true" or:origin="or:intended">
      ianaift:ethernetCsmacd
    </type>
    <mtu>1500</mtu>
  </interface>
</interfaces>
```

A.2. Built-in Access Control Example

In this example, the "ietf-netconf-acm" YANG module defined in [RFC8341] is used. Suppose when the device is powered on, the system may provide some built-in access control groups, for each access control group there exists at least one built-in user, which might be read-only and cannot be modified by the client. In addition, the system also can predefine some access control rules for a specific group. Some rules can be modified, while others are not (e.g., access control rule for a root account should be immutable). In addition, clients can also define their own groups and access control rules freely.

The built-in access control groups and rules are provided by the system, they are present in <operational>. The example below

illustrates what the built-in groups and rules might look like for a server after it boots:

```
<nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-acm"
      xmlns:im="urn:ietf:params:xml:ns:yang:ietf-immutable">
  <groups>
    <group>
      <name>admin</name>
      <user-name im:immutable="true">admin</user-name>
    </group>
    <group>
      <name>monitor</name>
      <user-name im:immutable="true">user</user-name>
    </group>
    <group>
      <name>visit</name>
      <user-name im:immutable="true">guest</user-name>
    </group>
  </groups>
  <rule-list im:immutable="true">
    <name>admin-acl</name>
    <group>admin</group>
    <rule>
      <name>permit-all</name>
      <module-name>*</module-name>
      <access-operations>*</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
  <rule-list>
    <name>monitor-acl</name>
    <group>visit</group>
    <rule>
      <name>permit-monitor</name>
      <module-name>ietf-netconf-monitoring</module-name>
      <access-operations>read</access-operations>
      <action>permit</action>
    </rule>
  </rule-list>
  <rule-list>
    <name>visit-acl</name>
    <group>visit</group>
    <rule>
      <name>deny-ncm</name>
      <module-name>ietf-netconf-monitoring</module-name>
      <access-operations>*</access-operations>
      <action>deny</action>
    </rule>
  </rule-list>
</nacm>
```

As indicated in the above XML snippet, the system predefines three access control groups and for each group there is one built-in immutable user, which means that the client can define new users for a particular access control group, but deletion of a built-in user-name is not allowed. The system also provides three access control list entries, one for each predefined group. Only the access control rule for the "admin" group is read-only to clients.

Authors' Addresses

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

Email: maqiufang1@huawei.com

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

Email: bill.wu@huawei.com

Hongwei Li
HPE

Email: flycoolman@gmail.com