

NETMOD  
Internet-Draft  
Intended status: Standards Track  
Expires: 24 April 2024

Q. Ma  
Q. Wu  
Huawei  
B. Lengyel  
Ericsson  
H. Li  
HPE

22 October 2023

YANG Metadata Annotation for Immutable Flag  
draft-ma-netmod-immutable-flag-09

Abstract

This document defines a way to formally document existing behavior, implemented by servers in production, on the immutability of some system configuration nodes, using a YANG metadata annotation called "immutable" to flag which nodes are immutable.

Clients may use "immutable" annotations provided by the server, to know beforehand why certain otherwise valid configuration requests will cause the server to return an error.

The immutable flag is descriptive, documenting existing behavior, not proscriptive, dictating server behavior.

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 24 April 2024.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in 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 . . . . .	3
1.1. Terminology . . . . .	4
1.2. Applicability . . . . .	5
2. Solution Overview . . . . .	5
3. "Immutable" Metadata Annotation . . . . .	5
3.1. Definition . . . . .	6
3.2. "with-immutable" Parameter . . . . .	6
4. Use of "immutable" Flag for Different Statements . . . . .	6
4.1. The "leaf" Statement . . . . .	7
4.2. The "leaf-list" Statement . . . . .	7
4.3. The "container" Statement . . . . .	7
4.4. The "list" Statement . . . . .	7
4.5. The "anydata" Statement . . . . .	7
4.6. The "anyxml" Statement . . . . .	7
5. Immutability of Interior Nodes . . . . .	8
6. Interaction between Immutable Flag and <system> . . . . .	8
7. Interaction between Immutable Flag and NACM . . . . .	9
8. YANG Module . . . . .	9
9. IANA Considerations . . . . .	11
9.1. The "IETF XML" Registry . . . . .	12
9.2. The "YANG Module Names" Registry . . . . .	12
10. Security Considerations . . . . .	12
Acknowledgements . . . . .	12
References . . . . .	12
Normative References . . . . .	12
Informative References . . . . .	13
Appendix A. Detailed Use Cases . . . . .	14
A.1. UC1 - Modeling of server capabilities . . . . .	14
A.2. UC2 - HW based auto-configuration - Interface Example . .	15
A.3. UC3 - Predefined Administrator Roles . . . . .	15
A.4. UC4 - Declaring immutable system configuration from an LNE's perspective . . . . .	16
Appendix B. Existing implementations . . . . .	16
Appendix C. Changes between revisions . . . . .	16
Appendix D. Open Issues tracking . . . . .	19
Authors' Addresses . . . . .	19

## 1. Introduction

This document defines a way to formally document as a YANG metadata annotation an existing model handling behavior that has been used by multiple standard organizations and vendors. It is the aim to create one single standard solution for documenting non-modifiable system data declared as configuration, instead of the multiple existing vendor and organization specific solutions. See Appendix B for existing implementations.

YANG [RFC7950] is a data modeling language used to model both state and configuration data, based on the "config" statement. However, there exists some system configuration data that cannot be modified by the client (it is immutable), but still needs to be declared as "config true" to:

- \* allow configuration of data nodes under immutable lists or containers;
- \* place "when", "must" and "leafref" constraints between configuration and immutable data nodes.
- \* ensure the existence of specific list entries that are provided and needed by the system, while additional list entries can be created, modified or deleted;

If the server always rejects the client attempts to override immutable system configuration [I-D.ietf-netmod-system-config] because it internally thinks it immutable, it should document this towards the clients in a machine-readable way rather than writing as plain text in the description statement.

This document defines a way to formally document existing behavior, implemented by servers in production, on the immutability of some system configuration nodes, using a YANG metadata annotation [RFC7952] called "immutable" to flag which nodes are immutable.

This document does not apply to the server not having any immutable system configuration. While in some cases immutability may be needed, it also has disadvantages, therefore it SHOULD be avoided wherever possible.

The following is a list of already implemented and potential use cases.

UC1 Modeling of server capabilities

UC2 HW based auto-configuration

UC3 Predefined administrator roles

UC4 Declaring immutable system configuration from an LNE's perspective

Appendix A describes the use cases in detail.

### 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]:

- \* configuration data

The following terms are defined in [RFC7950]:

- \* data node

- \* leaf

- \* leaf-list

- \* container

- \* list

- \* anydata

- \* anyxml

- \* interior node

- \* data tree

The following terms are defined in [RFC8341]:

- \* access operation

- \* write access

The following terms are defined in this document:

immutable flag: A read-only state value the server provides to

describe system data it considers immutable. The immutability of data nodes is conveyed via a YANG metadata annotation called "immutable".

## 1.2. Applicability

This document focuses on the configuration which can only be created, updated and deleted by the server.

The immutable annotation information is also visible in read-only datastores like <system> (if exists), <intended> and <operational> when a "with-immutable" parameter is carried (see Section 3.2), however this only serves as descriptive information about the instance node itself, but has no effect on the handling of the read-only datastore.

Configuration data must have the same immutability in different writable datastores. The immutability of data nodes is protocol and user independent. The immutability and configured value of an existing node must only change by software upgrade or hardware resource/license change.

## 2. Solution Overview

Immutable configuration can only be created by the system regardless of the implementation of <system> [I-D.ietf-netmod-system-config]. Immutable configuration is present in <system> (if implements). It may be updated or deleted depending on factors like software upgrade or hardware resources/license change. Immutable configuration does not appear in <running> unless it is copied explicitly or automatically (e.g., by "resolve-system" parameter) [I-D.ietf-netmod-system-config].

A client may create/delete immutable nodes with same values as found in <system> (if exists) in read-write configuration datastore (e.g., <running>), which merely mean making immutable nodes visible/invisible in read-write configuration datastore (e.g., <running>).

The "immutable" flag is intended to be descriptive.

## 3. "Immutable" Metadata Annotation

### 3.1. Definition

The "immutable" metadata annotation takes as an value which is a boolean type, it is not returned unless a client explicitly requests through a "with-immutable" parameter (see Section 3.2). If the "immutable" metadata annotation for data node instances is not specified, the default "immutable" value is the same as the immutability of its parent node in the data tree. The immutable metadata annotation value for a top-level instance node is false if not specified.

Note that "immutable" metadata annotation is used to annotate data node instances. A list may have multiple entries/instances in the data tree, "immutable" can annotate some of the instances as read-only, while others are read-write.

### 3.2. "with-immutable" Parameter

The YANG model defined in this document (see Section 8) augments the <get-config>, <get> operation defined in RFC 6241, and the <get-data> operation defined in RFC 8526 with a new parameter named "with-immutable". When this parameter is present, it requests that the server includes "immutable" metadata annotations in its response.

This parameter may be used for read-only configuration datastores, e.g., <system> (if exists), <intended> and <operational>, but the "immutable" metadata annotation returned indicates the immutability towards read-write configuration datastores, e.g., <startup>, <candidate> and <running>. If the "immutable" metadata annotation for returned child nodes are omitted, it has the same immutability as its parent node. The immutability of top hierarchy of returned nodes is false by default.

Note that "immutable" metadata annotation is not included in a response unless a client explicitly requests them with a "with-immutable" parameter.

## 4. Use of "immutable" Flag for Different Statements

This section defines what the immutable flag means to the client for each instance of YANG data node statement.

Throughout this section, the word "change" refers to create, update, and delete.

#### 4.1. The "leaf" Statement

When a leaf node instance is immutable, its value cannot change.

#### 4.2. The "leaf-list" Statement

When a leaf-list node instance is immutable, its value cannot change.

When the "immutable" YANG metadata annotation is used on all existing leaf-list instances, or if a leaf-list inherits immutability from an ancestor, it means that the leaf-list as a whole cannot change: entries cannot be added, removed, or reordered, in case the leaf-list is "ordered-by user".

#### 4.3. The "container" Statement

When a container node instance is immutable, it cannot change, unless the immutability of its descendant node is toggled.

By default, as with all interior nodes, immutability is recursively applied to descendants (see Section 5).

#### 4.4. The "list" Statement

When a list node instance is immutable, it cannot change, unless the immutability of its descendant node is toggled, per the description elsewhere in this section.

By default, as with all interior nodes, immutability is recursively applied to descendants (see Section 5). This statement is applicable only to the "immutable" YANG extension, as the "list" node does not itself appear in data trees.

#### 4.5. The "anydata" Statement

When an anydata node instance is immutable, it cannot change. Additionally, as with all interior nodes, immutability is recursively applied to descendants (see Section 5).

#### 4.6. The "anyxml" Statement

When an "anyxml" node instance is immutable, it cannot change. Additionally, as with all interior nodes, immutability is recursively applied to descendants (see Section 5).

## 5. Immutability of Interior Nodes

Immutability is a conceptual operational state value that is recursively applied to descendants, which may reset the immutability state as needed, thereby affecting their descendants. There is no limit to the number of times the immutability state may change in a data tree.

For example, given the following application configuration XML snippets:

```
<application im:immutable="true">
  <name>predefined-ftp</name>
  <protocol>ftp</protocol>
  <port-number im:immutable="false">69</port-number>
</application>
```

The list entry named "predefined-ftp" is immutable="true", but its child node "port-number" has the immutable="false" (thus the client can override this value). The other child node (e.g., "protocol") not specifying its immutability explicitly inherits immutability from its parent node thus is also immutable="true".

## 6. Interaction between Immutable Flag and <system>

The system datastore is defined to hold system configuration provided by the device itself and make system configuration visible to clients in order for being referenced or configurable prior to present in <operational>. However, the device may allow some system-initialized node to be overridden, while others may not. System configuration exists regardless of whether <system> is implemented.

This document defines a way to allow a server annotate instances of non-modifiable system configuration with metadata when system configuration is retrieved. A client aware of the "immutable" annotation can explicitly ask the server to return it via the "with-immutable" parameter in the request, thus is able to avoid making unnecessary modification attempts to immutable configuration. Legacy clients unaware of the "immutable" annotation don't see any changes and encounter an error as always.



## 7. Interaction between Immutable Flag and NACM

The server rejects an operation request due to immutability when it tries to perform the operation on the request data. It happens after any access control processing, if the Network Configuration Access Control Model (NACM) [RFC8341] is implemented on a server. For example, if an operation requests to override an immutable configuration data, but the server checks the user is not authorized to perform the requested access operation on the request data, the request is rejected with an "access-denied" error.

## 8. YANG Module

```
<CODE BEGINS>
file="ietf-immutable@2023-10-16.yang"
//RFC Ed.: replace XXXX with RFC number and remove this note
module ietf-immutable {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-immutable";
  prefix im;

  import ietf-yang-metadata {
    prefix md;
  }
  import ietf-netconf {
    prefix nc;
    reference
      "RFC 6241: Network Configuration Protocol (NETCONF)";
  }
  import ietf-netconf-nmda {
    prefix ncds;
    reference
      "RFC 8526: NETCONF Extensions to Support the Network
      Management Datastore Architecture";
  }
  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: Balazs Lengyel  
<mailto:balazs.lengyel@ericsson.com>

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

description

"This module defines a metadata annotation called 'immutable' to allow the server to formally document existing behavior on the mutability of some system configuration. Clients may use 'immutable' metadata annotation provided by the server to know beforehand why certain otherwise valid configuration requests will cause the server to return an error.

Copyright (c) 2023 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 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 2023-10-16 {  
  description  
    "Initial revision."  
  // RFC Ed.: replace XXXX and remove this comment  
  reference  
    "RFC XXXX: YANG Metadata Annotation for Immutable Flag";  
}
```

```
md:annotation immutable {  
  type boolean;  
  description  
    "The 'immutable' metadata annotation indicates the  
    immutability of an instantiated data node.
```

```
    The 'immutable' metadata annotation takes as a value 'true'
    or 'false'. If the 'immutable' metadata annotation for data
    node instances is not specified, the default value is the
    same as the value of its parent node in the data tree. The
    default value for a top-level instance node is false if not
    specified.";
  }

  grouping with-immutable-grouping {
    description
      "Grouping for the with-immutable parameter that augments the
      RPC operations.";
    leaf with-immutable {
      type empty;
      description
        "If this parameter is present, the server will return the
        'immutable' annotation for configuration that it
        internally thinks it immutable. When present, this
        parameter allows the server to formally document existing
        behavior on the mutability of some configuration nodes.";
    }
  }
}
augment "/ncds:get-data/ncds:input" {
  description
    "Allows the server to include 'immutable' metadata
    annotations in its response to get-data operation.";
  uses with-immutable-grouping;
}
augment "/nc:get-config/nc:input" {
  description
    "Allows the server to include 'immutable' metadata
    annotations in its response to get-config operation.";
  uses with-immutable-grouping;
}
augment "/nc:get/nc:input" {
  description
    "Allows the server to include 'immutable' metadata
    annotations in its response to get operation.";
  uses with-immutable-grouping;
}
}
<CODE ENDS>
```

## 9. IANA Considerations

### 9.1. 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.

### 9.2. 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

## 10. Security Considerations

The YANG module specified in this document defines a YANG extension and a metadata Annotation. These can be used to further restrict write access but cannot be used to extend access rights.

This document does not define any protocol-accessible data nodes.

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.

## Acknowledgements

Thanks to Kent Watsen, Andy Bierman, Robert Wilton, Jan Lindblad, Reshad Rahman, Anthony Somerset, Lou Berger, Joe Clarke, Scott Mansfield, and Juergen Schoenwaelder for reviewing, and providing important inputs to, this document.

## References

### 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>>.
- [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>>.
- [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>>.
- [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>>.

#### Informative References

- [I-D.ietf-netmod-system-config] Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-03, 19 October 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-03>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [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>>.
- [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>>.

- [RFC8530] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Model for Logical Network Elements", RFC 8530, DOI 10.17487/RFC8530, March 2019, <<https://www.rfc-editor.org/info/rfc8530>>.
- [TR-531] ONF, "UML to YANG Mapping Guidelines, <[https://wiki.opennetworking.org/download/attachments/376340494/Draft\\_TR-531\\_UML-YANG\\_Mapping\\_Gdls\\_v1.1.03.docx?version=5&modificationDate=1675432243513&api=v2](https://wiki.opennetworking.org/download/attachments/376340494/Draft_TR-531_UML-YANG_Mapping_Gdls_v1.1.03.docx?version=5&modificationDate=1675432243513&api=v2)>", February 2023.
- [TS28.623] 3GPP, "Telecommunication management; Generic Network Resource Model (NRM) Integration Reference Point (IRP); Solution Set (SS) definitions, <[https://www.3gpp.org/ftp/Specs/archive/28\\_series/28.623/28623-i02.zip](https://www.3gpp.org/ftp/Specs/archive/28_series/28.623/28623-i02.zip)>".
- [TS32.156] 3GPP, "Telecommunication management; Fixed Mobile Convergence (FMC) Model repertoire, <[https://www.3gpp.org/ftp/Specs/archive/32\\_series/32.156/32156-h10.zip](https://www.3gpp.org/ftp/Specs/archive/32_series/32.156/32156-h10.zip)>".

## Appendix A. Detailed Use Cases

### A.1. UC1 - Modeling of server capabilities

System capabilities might be represented as system-defined data nodes in the model. Configurable data nodes might need constraints specified as "when", "must" or "path" statements to ensure that configuration is set according to the system's capabilities. E.g.,

- \* A timer can support the values 1,5,8 seconds. This is defined in the leaf-list 'supported-timer-values'.
- \* When the configurable 'interface-timer' leaf is set, it should be ensured that one of the supported values is used. The natural solution would be to make the 'interface-timer' a leaf-ref pointing at the 'supported-timer-values'.

However, this is not possible as 'supported-timer-values' must be read-only thus config=false while 'interface-timer' must be writable thus config=true. According to the rules of YANG it is not allowed to put a constraint between config true and false data nodes.

The solution is that the supported-timer-values data node in the YANG Model shall be defined as "config true" and shall also be marked with the "immutable" extension making it unchangeable. After this the 'interface-timer' shall be defined as a leaf-ref pointing at the 'supported-timer-values'.

#### A.2. UC2 - HW based auto-configuration - Interface Example

[RFC8343] defines a YANG data model for the management of network interfaces. When a system-controlled interface is physically present, the system creates an interface entry with valid name and type values in <system> (if exists, see [I-D.ietf-netmod-system-config]).

The system-generated type value is dependent on and represents the HW present, and as a consequence cannot be changed by the client. If a client tries to set the type of an interface to a value that can never be used by the system, the request will be rejected by the server. The data is modelled as "config true" and should be annotated as immutable.

Seemingly an alternative would be to model the list and these leaves as "config false", but that does not work because:

- \* The list cannot be marked as "config false", because it needs to contain configurable child nodes, e.g., ip-address or enabled;
- \* The key leaf (name) cannot be marked as "config false" as the list itself is config true;
- \* The type cannot be marked "config false", because we MAY need to reference the type to make different configuration nodes conditionally available.

#### A.3. UC3 - Predefined Administrator Roles

User and group management is fundamental for setting up access control rules (see section 2.5 of [RFC8341]).

A device may provide a predefined user account (e.g., a system administrator that is always available and has full privileges) for initial system set up and management of other users/groups. It is possible that clients can define a new user/group and grant it particular privileges, but the predefined administrator account and its granted access cannot be modified.

#### A.4. UC4 - Declaring immutable system configuration from an LNE's perspective

An LNE (logical network element) is an independently managed virtual network device made up of resources allocated to it from its host or parent network device [RFC8530]. The host device may allocate some resources to an LNE, which from an LNE's perspective is provided by the system and may not be modifiable.

For example, a host may allocate an interface to an LNE with a valid MTU value as its management interface, so that the allocated interface should then be accessible as the LNE-specific instance of the interface model. The assigned MTU value is system-created and immutable from the context of the LNE.

#### Appendix B. Existing implementations

There are already a number of full or partial implementations of immutability.

3GPP TS 32.156 [TS32.156] and 28.623 [TS28.623]: Requirements and a partial solution

ITU-T using ONF TR-531[TR-531] concept on information model level but no YANG representation.

Ericsson: requirements and solution

YumaPro: requirements and solution

Nokia: partial requirements and solution

Huawei: partial requirements and solution

Cisco using the concept at least in some YANG modules

Junos OS provides a hidden and immutable configuration group called junos-defaults

#### Appendix C. Changes between revisions

Note to RFC Editor (To be removed by RFC Editor)

v08 - v09

- \* Remove immutable YANG extension definition to simplify the solution



- \* Add a new section to discuss the interaction between immutable flag and <system>
- \* Remove the error response example in Appendix A.
- \* rewrite UC3, rename it to "Predefined Administrator Roles"

## v06 - v07

- \* Use a Boolean type for the immutable value in YANG extension and metadata annotation
- \* Define a "with-immutable" parameter and state that immutable metadata annotation is not included in a response unless a client explicitly requests them with a "with-immutable" parameter
- \* reword the abstract and related introduction section to highlight immutable flag is descriptive
- \* Add a new section to define immutability of interior nodes, and merge with "Inheritance of Immutable configuration" section
- \* Add a new section to define what the immutable flag means for each YANG data node
- \* Define the "immutable flag" term.
- \* Add an item in the open issues tracking: Should the "immutable" metadata annotation also be returned for nodes described as immutable in the YANG schema so that there is a single source of truth?

## v05 - v06

- \* Remove immutable BGP AS number case
- \* Fix nits

## v04 - v05

- \* Emphasized that the proposal tries to formally document existing allowed behavior
- \* Reword the abstract and introduction sections;
- \* Restructure the document;
- \* Simplified the interface example in Appendix;

- \* Add immutable BGP AS number and peer-type configuration example.
- \* Added temporary section in Appendix B about list of existing non-standard solutions
- \* Clarified inheritance of immutability
- \* Clarified that this draft is not dependent on the existence of the <system> datastore.

v03 - v04

- \* Clarify how immutable flag interacts with NACM mechanism.

v02 - v03

- \* rephrase and avoid using "server MUST reject" statement, and try to clarify that this documents aims to provide visibility into existing immutable behavior;
- \* Add a new section to discuss the inheritance of immutability;
- \* Clarify that deletion to an immutable node in <running> which is instantiated in <system> and copied into <running> should always be allowed;
- \* Clarify that write access restriction due to general YANG rules has no need to be marked as immutable.
- \* Add an new section named "Acknowledgements";
- \* editorial changes.

v01 - v02

- \* clarify the relation between the creation/deletion of the immutable data node with its parent data node;
- \* Add a "TODO" comment about the inheritance of the immutable property;
- \* Define that the server should reject write attempt to the immutable data node at an <edit-config> operation time, rather than waiting until a <commit> or <validate> operation takes place;

v00 - v01

- \* Added immutable extension

- \* Added new use-cases for immutable extension and annotation
- \* Added requirement that an update that means no effective change should always be allowed
- \* Added clarification that immutable is only applied to read-write datastore
- \* Narrowed the applied scope of metadata annotation to list/leaf-list instances

#### Appendix D. Open Issues tracking

- \* Is this needed: error-code definition for edit failure because of immutability

#### Authors' Addresses

Qiufang Ma  
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

Balazs Lengyel  
Ericsson  
Email: balazs.lengyel@ericsson.com

Hongwei Li  
HPE  
Email: flycoolman@gmail.com