YANG Extension and Metadata Annotation for Immutable Flag

draft-ma-netmod-immutable-flag-02

Qiufang Ma (Huawei) Presenter
Qin Wu (Huawei)
Balazs Lengyel (Ericsson)
Hongwei Li (HPE)
Use Cases

Some "config true" data nodes are immutable independent of how it is instantiated

Some data node may exist in multiple instances in the data tree, some of which are immutable, which others are not
Motivation and Solution

• It is already the case today:
  • a server can reject any configuration for any reason;
  • e.g., when a client is trying to modify an immutable configuration;
  • This work provides more visibility to the client as to which nodes are immutable.

• Solution preferred:
  • “immutable” YANG extension
    • A substatement to a leaf, leaf-list, container, list, anydata or anyxml statement
    • An argument statement named “exceptions” is also defined to indicate specific operations (create, update, delete) are permitted
  • “immutable” Metadata annotation
    • Once a particular instantiated data node is created, the client cannot update/delete it
    • Be applied to list/leaf-list entries or instances inside particular list entries
    • Boolean values to indicate whether the node instance is immutable or not
  • Statement: the server should not return annotation if a particular data node is already marked as immutable by YANG extension without exceptions for “update” or “delete” operations.
Open Issue: is the preferred solution best?

Alternatives(1/3)

• Metadata annotation only (adopted in the v-00)

  • Pros:
    • Single solution
    • Immutable annotation is returned with associated retrieved data

  • Cons:
    • attach to existing instances, unable to convey : “creation NOT allowed”
    • Hard to understand if “immutable” is applied to the whole list, the list entries, or both; same applies to the leaf-list
    • Only available in runtime

[XML code]

```xml
  <interface>
    <name>lo0</name>
    <type im:immutable="true">ianaift:softwareloopback</type>
    <mtu>1500</mtu>
  </interface>
  <interface>
    <name>Ge0/0/0</name>
    <type im:immutable="true">ianaift:ethernetCsmacd</type>
    <mtu>1500</mtu>
  </interface>
</interfaces>
```

An immutable annotation usage example
Open Issue: is the preferred solution best?

Alternatives (2/3)

• NACM

  • Possible solutions:
    • Reuse NAC rules: i.e., system-defined NAC rules to immutable nodes;
    • Define a nacm:immutable extension besides nacm:default-deny-write and nacm:default-deny-all

  • Pro:
    • Reusing NAC rules is a single solution for nodes both in the schema tree and in the data tree

  • Cons:
    • NACM can be disabled by setting the “enable-nacm” leaf to “false”
    • Emergency recovery session will bypass access control enforcement
    • NAC rules are also immutable and how to protect them from being modified
      • E.g., when a new rule added before other existing rules is matched, all rules behind this new rule will be neglected.
    • A single extension cannot express instance-level immutability
Open Issue: is the preferred solution best?

Alternatives(3/3)

• Node-tags
  • Schema-level immutability can use a registered “ntags:node-tag ‘ietf:immutable’” extension
  • Pro:
    • single solution for nodes both in the schema tree and in the data tree
  • Cons:
    • Immutable tag is returned by reading a separate “node-tags” YANG model
    • Tags info for immutability is also immutable and how to protect it from being removed

```
<module-tags>
  <module>
    <name>example-applications</name>
    <node-tags>
      <node>
        <id>/applications/application[application-name="ssh"]</id>
        <tags>
          <tag>ietf:immutable</tag>
        </tags>
      </node>
    </node-tags>
  </module>
  ...
</module>
</module-tags>
```

A instance-level immutable node-tag usage example
Comments, Questions, Concerns?