IETF NETMOD Working Group Virtual Interim

System-defined configuration

Current: draft-ma-netconf-with-system-02 (stale)
Pending: draft-ma-netmod-with-system-00

Much data collected on mailing list

- 40+ number of messages
- Thank you! (in order of appearance):
  - Qiufang, Balazs, Andy, Fengchong (frank), Juergen, Jason, Qin, and Jan.

- Notes taken while re-reading the entire thread.
  - This presentation represents many viewpoints given.
    - Some text copy/pasted from thread

Objectives

- Enable systems to better document that which they do already. A standard mechanism to see what system config is available in a server.
- Avoid having to copy system configuration into <running> when possible.
- Support configuration of descendant nodes of system-defined nodes.
  - The well-known "interface problem". The system configures almost empty physical interfaces. The user is allowed to add, modify, and delete all descendants except the "name" and "type" leaf, which are set by the system.
- A read-back of <running> SHOULD contain just what was set by clients
  - Known exception: RFC 7317’s ianach:crypt-hash for passwords:

```
The '$0$' prefix signals that the value is clear text.  When
such a value is received by the server, a hash value is
calculated, and the string '$<id>$<salt>$' or
$<id>$<parameter>$<salt>$ is prepended to the result.  This
value is stored in the configuration data store.
```

Axes of Interest

- **<u>Inactive-until-referenced</u> system-defined nodes** (e.g., predefined objects)
  - e.g., application ids, anti-x signatures, trust anchor certs, etc.

- **<u>Immediately-active</u> system-defined configuration** (e.g., loopback, eth0, etc.)
  - not in <factory default> as system-config isn't deletable (possible "immutable" extension?)
  - not in <operational> only, because some system-defined nodes need to be reference-able.

- **<u>Conditionally-active</u> system-defined configuration** (e.g., pluggable cards, etc.)
  - Like "when" expressions, whereby the condition is on system resources…
  - why not in <operational> w/ origin=system? (PRO: dynamic change already expected)
    - Because some subnodes configurable.

System change outside software upgrade?

[Regarding: conditionally-active system-defined configuration (e.g., pluggable cards, etc.)]

Comments on list:

- “QOS function is enabled, many qos predefined policies are created by system”
- “I agree there can be dynamically added system config (e.g. create a new qos policy, and some queue list entries are automatically created inside that policy).”

If system-config can change, how to notify client?

- YANG notification

System-defined Nodes Are Not Deletable

Is this true?

Any **deletable** “system” defined node can be defined in <factory-default>

- <factory-default> nodes are already deletable.
- Any system-defined ancestor nodes (e.g., eth0’s parents) must be copied.

Corollary: Any **deletable** node is **modifiable**

- Client can always delete and recreate, so totally modifiable.
- No need to limit modifiability of **deletable**-nodes and/or their descendants.

System-defined Nodes MAY be Modifiable

Descendent node addition/modification may be allowed.

- This why cannot be in <factory-default> (unless there were an “immutable” flag)

Example of a modifiable, but not deletable, node:

- A “mandatory true” leaf with a system-specified default value.

No impact to <operational>

- As always, system-defined nodes appear in <operational> w/ origin=“system”.

- This work enables a subset of those nodes to be defined like config.
  - Yet they still appear with origin=“system”

- Existing “config false” nodes are not impacted by this work.

Solution Considerations

Solution Considerations

1. **Use <factory-defaults>** (not enough)
   - Good for deletable system-defined config, but most system config is not deletable *(until an "immutable" flag is defined)*
   - Also, <factory-defaults> is not expected to change based on resource conditions.

2. **Use <operational>** (not enough)
   - Yes, all **active** system config is present
   - But unreferenced shared objects would have no impact on data plane and hence may not be present in <operational>
   - And some nodes need to be referenced by config in <running>

Solution Considerations (cont.)

3. **Use <running>** *(three variations below)*

   a) Copy from <operational> via explicit client action (<edit-config>)

      - But only active config in <operational>, right? (Catch-22) (not enough)

   b) Implicitly hidden nodes, two options:

      i. Implicitly hidden, made visible via a “with-system” parameter (seems viable)

      ii. Hidden by NACM, made visible via proper authorization (NACM not mandatory)

Solution Considerations (cont.)

4. Use <system> (seems viable)

- Standard NMDA datastore access to all “config true” nodes
- Limited to read-only RPCs, but datastore content not static:
  - Content may change by upgrades and/or when resource-conditions are met
    - A YANG ‘notification’ is needed. *(Just <system> or any datastore changed?)*
- Offline validation necessitates clients understand how to merge
  - Workaround: use the <validate> RPC, if supported.

Solution Considerations (cont.)

4. Use <system> (cont.)

Concern: Debugging a configuration distributed over multiple datastores is difficult.

Considerations:

- Non-issue for config copied into <running>.
- For mis-referenced objects, validation errors should explain.
- Servers can present a merged/expanded <intended> datastore
  - For clients that don’t want to understand the merge logic.

Solution Considerations (cont.)

- Both 3b + optionally 4?

Servers MUST support a “with-system” parameter

AND

Servers MAY support a read-only <system> datastore

Any server supporting <system> SHOULD support an RPC to get a merged view, right?

Solution Considerations (cont.)

- What is this “immutable” flag idea about?
- The idea is to define per-node metadata flag (using an RFC 7952 annotation) called “immutable”
- Any node marked with the flag by the server is read-only to clients
- This solution enables, e.g., a host-system to share resources with logical systems (i.e., RFC 8530 logical network elements, LNEs) that are read-only from an LNE’s perspective…
- Implementation of this idea may require YANG-Next….

Example from JUNOS

JUNOS Example: Shared Objects

- In /<running>/groups/junos-defaults *(but could also be in <system>)*

```
system-defined-defaults {
        applications {
                application ftp {
                        protocol tcp;
                        destination-port 21;
                }
                application smtp {
                        protocol tcp;
                        destination-port 25;
                }
                ...
         }
}
```

- NOTE: junos-defaults is hidden in JUNOS

Example: Shared Objects (cont.)

- And also this in /<running>/:

```
// custom objects use the same schema
applications {
        application my-app-1 {
                protocol tcp;
                 destination-port 2345;
        }
 }

// an ACL policy referencing both sys-defined and custom objects
policy from-zone untrust to-zone untrust {
        policy allow-external-access-to-foobar-app {
                match {
                        source-address any;
                        destination-address any;
                        application [ ftp tftp, my-app-1 ];
                }
                then {
                        permit;
                }
        }
}
```