RESTCONF with Transactions

`draft-lhotka-netconf-restconf-transactions-00`

Ladislav Lhotka
⟨lhotka@nic.cz⟩

16 July 2018

Objectives

- transactions with explicit client’s control
- concurrent R/W access of multiple clients
- simple enhancement of RFC 8040, maximum backward compatibility
- NMDA-compliant

Server Implementation

Three datastores:

- operational (NMDA)
- intended (NMDA, but persists across reboots)
- staging (configuration, “per-user private candidate”)

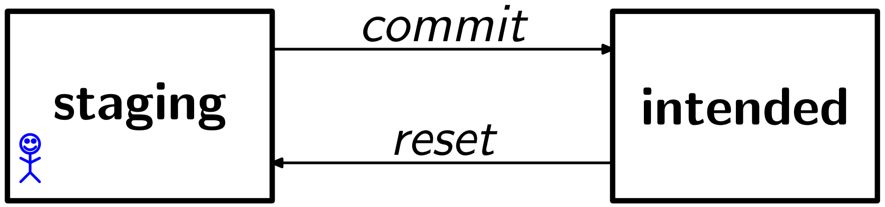
⟨staging⟩ assumes the role of the “unified” RESTCONF datastore:

```
PUT /restconf/data/example-jukebox:jukebox/\
    library/artist=Foo%20Fighters/album=Wasting%20Light
```

Resources corresponding to ⟨intended⟩ and ⟨operational⟩ are defined in *draft-ietf-netconf-nmda-restconf*.

Recommended implementation of config datastores:
persistent data structures with copy-on-write.

New Operations



commit: merge ⟨staging⟩ atomically into ⟨intended⟩

reset: reset ⟨staging⟩ to the content of ⟨intended⟩

Requirements:

- ⟨intended⟩ must always be valid
- after both operations, ⟨staging⟩ and ⟨intended⟩ must have (conceptually) the same content

Merge Procedure

Left intentionally unspecified: different use cases may need different approaches.

Merge conflicts should be preferably resolved automatically, it is also possible that the client be asked for a manual intervention.

Compatibility with RFC 8040

The presence of ⟨staging⟩ is *almost* transparent to the user: the interaction is the same as with standard RESTCONF, except that configuration changes are not applied.

Clients supporting standard RESTCONF can be used for reading datastores and editing ⟨staging⟩, the *commit* and *reset* operations can be provided separately (e.g. as curl scripts).

Naming Issues

Different datastores (names) were suggested:

$$\begin{aligned} \text{staging} &\rightarrow \text{candidate} \\ \text{intended} &\rightarrow \text{running} \end{aligned}$$

- properties of ⟨intended⟩ are close to what we need (read-only, always valid)
- the datastores on the right are used in NETCONF and their semantics is (may be) incompatible: writable ⟨running⟩, shared ⟨candidate⟩
- little interference if NETCONF is used on the same device: contributions from RESTCONF and NETCONF come together in ⟨intended⟩

Running Code

JetConf: https://github.com/CZ-NIC/jetconf

- written in Python 3
- uses HTTP/2, only JSON representation
- client certificates for authentication
- callback API for writing specific back-ends
- *zipper* [1] structure used for configuration data
- not yet NMDA-compatible (datastores, resources)

[1] https://www.st.cs.uni-saarland.de/edu/seminare/2005/advanced-fp/docs/huet-zipper.pdf