NETCONF Transaction ID and related drafts

2023-03-27
IETF 116
Jan Lindblad <jlinldbla@cisco.com>
Roque Gagliano <rogaglia@cisco.com>
Transaction ID

draft-ietf-netconf-transaction-id-00
draft-ietf-netconf-transaction-id-00 changes (vs. draft-lindblad-...-02)

• Changed the logic around how txids are handled in the candidate datastore
• Changed the logic of copy-config to be similar to edit-config
• Clarified how txid values interact with when-dependencies together with default values

• Added content to security considerations
• Added a high-level example for YANG-Push subscriptions with txid
• Updated language about error-info sent at txid mismatch in an edit-config
• Some rewording and minor additions for clarification, based on mailing list feedback
New Txid Candidate Datastore Behavior

Consider a <get-config> towards :candidate. Most recent txid="3456"
   <get-config txid="?"/>

The :candidate may hold modified elements (vs. :running)
  • Some implementations may know before <commit> what the new txid would be if candidate was committed. They can return this, future, txid.  
    <data txid="4711">
      <some-leaf txid="4711">updated-value</some-leaf>
    </data>
  • Some implementations may not know. They can return txid "!". This indicates the txid will change at commit.
    <data txid="!">
      <some-leaf txid="!">updated-value</some-leaf>
    </data>
Next Steps

• Implementation experience
Trace Context Extension
draft-rogaglia-netconf-trace-ctx-extension-02
draft-rogaglia-netconf-trace-ctx-extension-02 changes (vs. -00)

• Added Implementation example 2: YANG DataStore to facilitate emphasize complementary with other drafts
• Added new use case: Billing and auditing
• Improved text related to: [I-D.ietf-netconf-transaction-id]
• Added Error Handling initial section
• Added how to manage versioning by defining YANG modules for each traceparent and tracestate versions as defined by W3C
• Added 'YANG Module Names' to IANA Considerations
Decorating RPCs with Trace IDs

Collector apps can:

- Correlate resource usage on all levels to individual customers and orders
- Trace the effects of service input through all layers, for billing, debugging or forensic purposes
Format Defined by W3C

https://www.w3.org/TR/2021/REC-trace-context-1-20211123/

Headers directly usable in RESTCONF:

traceparent: 00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01
tracestate: rojo=00f067aa0ba902b7,congo=t61rcWkgMzE

A NETCONF client might send this:

```xml
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1"
     xmlns:w3ctc="urn:ietf:params:xml:ns:netconf:w3ctc:1.0"
     w3ctc:tracestate="rojo=00f067aa0ba902b7,congo=t61rcWkgMzE"
     w3ctc:traceparent="00-4bf92f3577b34da6a3ce929d0e0e4736-00f067aa0ba902b7-01">
     <edit-config>…</edit-config>
</rpc>
```
W3C's got this

• Initially, trans-id-00 draft defined NETCONF attributes for this purpose (but only for config operations)
• Since then, W3C defined their HTTP trace headers for all operations (GET/PUT/PATCH/POST/DELETE)
• W3C have tracing solution vendors on board

♫ We expect straight-forward integration with mainstream tools if we base our solution on W3C's headers and encoding
The end goal is to tap into OTLP ecosystem
Example with ServiceNow/LightStep backend

Orchestrator

<edit-config> trace-context

OTLP Backend

Controller

Orchestrator trace spans

Same trace-id

Controller trace spans
Objective: register NETCONF and RESTCONF at trace Context registry

ref: https://www.w3.org/TR/trace-context-protocols-registry/

§ 3. Registry

This section is the registry of identified formats of trace context [TRACE-CONTEXT] serialization and deserialization for protocols.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Public Specification(s)</th>
<th>Requestor Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>[TRACE-CONTEXT]</td>
<td>W3C</td>
</tr>
<tr>
<td>Binary</td>
<td>Trace Context: binary protocol</td>
<td>W3C</td>
</tr>
<tr>
<td>[AMQP]</td>
<td>Trace Context: AMQP protocol</td>
<td>W3C</td>
</tr>
<tr>
<td>[MQTT]</td>
<td>Trace Context: MQTT protocol</td>
<td>W3C</td>
</tr>
</tbody>
</table>

Add RESTCONF and NETCONF to this list!
Next Steps

- More details around error handling
- Publish RESTCONF document “should” implement W3C headers
- Publish “Baggage” draft
- Figure out process for W3C protocol registration
- Implementation experience
Overview of drafts related to Transaction ID
# Overview of Related Drafts

<table>
<thead>
<tr>
<th>Draft ID</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANS-ID</td>
<td>draft-ietf-netconf-transaction-id-00</td>
</tr>
<tr>
<td>CFG-TRACE</td>
<td>draft-quilbeuf-opsawg-configuration-tracing-01</td>
</tr>
<tr>
<td>W3C-TRACE</td>
<td>draft-rogaglia-netconf-trace-ctx-extension-02</td>
</tr>
<tr>
<td>PRIV-CAND</td>
<td>draft-jgc-netconf-privcand-01 (not yet published)</td>
</tr>
<tr>
<td>ETAGS</td>
<td>RFC 8040 (RESTCONF)</td>
</tr>
</tbody>
</table>
## Massive Use Case Overlaps

* = Not in recent versions

<table>
<thead>
<tr>
<th>Use Case</th>
<th>TRANS-ID</th>
<th>CFG-TRACE</th>
<th>W3C-TRACE</th>
<th>PRIV-CAND</th>
<th>ETAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase transaction throughput by reducing lock time</td>
<td>✔</td>
<td>-</td>
<td>-</td>
<td>✔</td>
<td>-</td>
</tr>
<tr>
<td>Allow clients to get config changes at top level or within subtree (&quot;Sync&quot;)</td>
<td>✔</td>
<td>Assumed in other doc</td>
<td>-</td>
<td>-</td>
<td>✔</td>
</tr>
<tr>
<td>Allow clients to make config changes conditional on no conflicts (&quot;No overwrite&quot;)</td>
<td>✔</td>
<td>Assumed in other doc</td>
<td>-</td>
<td>Maybe</td>
<td>✔</td>
</tr>
<tr>
<td>Allow clients to recognize their own echo in YANG Push updates</td>
<td>✔</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Map transaction ids from client to server and server controlled children</td>
<td>*</td>
<td>✔</td>
<td>✔</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Finding Source of configuration mistakes</td>
<td>*</td>
<td>✔</td>
<td>✔</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Detecting conflicting intents</td>
<td>*</td>
<td>✔</td>
<td>✔</td>
<td>Maybe</td>
<td>-</td>
</tr>
<tr>
<td>Provisioning root cause analysis</td>
<td>*</td>
<td>✔</td>
<td>✔</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>System performance profiling</td>
<td>*</td>
<td>✔</td>
<td>✔</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Billing and auditing</td>
<td>*</td>
<td>✔</td>
<td>✔</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
... but not much overlap in functionality

Great overlap in use cases + not much overlap in functionality

=>

Drafts could be made into a coherent framework
Drafts are (mostly) Complementary

Orchestrator

Controller 1
- NE1

Controller 2
- NE2
- NE3
Drafts are (mostly) Complementary

Orchestrator

Controller 1

Controller 2

NE1

NE2

NE3

draft-jgc-netconf-privcand-00
Drafts are (mostly) Complementary

Orchestrator

Controller 1

Controller 2

NE1

NE2

NE3

draft-jgc-netconf-privcand-00

draft-ietf-netconf-transaction-id-00
Drafts are (mostly) Complementary

Orchestrator

Controller 1

Controller 2

NE1

NE2

NE3

M.E.L.T processor

draft-jgc-netconf-privcand-00

draft-ietf-netconf-transaction-id-00

draft-rogaglia-netconf-trace-ctx-extension-02

E.g. OTLP
Drafts are (mostly) Complementary

Orchestrator

Controller 1

Controller 2

NE1

NE2

NE3

YANG Txid Mapping

txid

priv

txid

M.E.L.T processor

draft-jgc-netconf-privcand-00

draft-ietf-netconf-transaction-id-00

draft-rogaglia-netconf-trace-ctx-extension-02

draft-quilbeuf-opsawg-configuration-tracing-01
Direction Going Forward (IMHO)

Create framework of 4 separate documents
• Each one optional to implement, optional to use
• Use cases, terminology and behavior aligned
Thank you