CoRE: CoRECONF, CRI

January 19th, (Wednesday), 15:00-16:30 UTC
(16:00–17:30 CET, 07:00–08:30 PST)

CoRECONF: 4 drafts

In IESG:

- YANG-CBOR: https://dt.ietf.org/doc/draft-ietf-core-yang-cbor/
- CORE-SID: https://dt.ietf.org/doc/draft-ietf-core-sid/

WGLC passed, waiting for writeup:

- COMI: https://dt.ietf.org/doc/draft-ietf-core-comi/
- CORE-YANG-LIBRARY: https://dt.ietf.org/doc/draft-ietf-core-yang-library/

CoRECONF in IESG

Plan: Do DISCUSS clearing first
(+ COMMENTS for DISCUSS-wielding ADs, as convenient)

yang-cbor-18: No DISCUSSes (RW).
Status: Should be "Revised I-D needed" (for COMMENTs)

core-sid-18: 1 DISCUSS remaining (RW).
Status: IESG Evaluation::AD Followup

CORE-SID

Rob Wilton keeps up one DISCUSS item:

— SID evolution/lifecycle

(Previous changes were insufficient to address issue)

SID evolution/lifecycle

Remaining DISCUSS item from Rob Wilton
(updated in 2022-01-07 mail):

How does the SID-name mapping evolve for a module?

— [x] Attach SID to item "semantics"

— [ ] Attach SID to item name

Note: All bets are off before publishing a module
(Some people think that this is RFC time — different issue)

Attach SID to "semantics"

Idea: Isolate implementations somewhat from name changes

But really: Why would one change a name in a published module?

Andy's explanation:

A SID to path mapping does depend on all the [module, revision-date] tuples used in the SID file. If the module uses groupings from other modules then those revision dates matter.
IMO if we can accept that YANG files can change in non-backward-compatible ways then the same has to be accepted for SID files.

Attach SID to name

Rob Wilton notes: […] I can foresee that [updating SID-name mappings] may make some interop issues trickier.

E.g., controller that understands two SID values for a given leaf. Configuring two devices, each of which only understands one of the SID values. The controller can easily be setup to receive data using either SID value, but when sending configuration would need to have a per client SID mapping.

[…] perhaps there should be a rule that clients MUST always accept the original published SID value? But this then places an ordering between SID definitions and would seem to make initial development SIDs permanent.

RW continued:

[…] I have a much bigger concern with conflating the SID with the data node definition/type, rather than the SID just being an alternative more efficient numeric representation of the (namespace, name) pair.
I think that it effectively means that your SID becomes a binding to (namespace, name, set of module revisions).
[…] doing this would be a mistake […]

RW:

- A) Should new SIDs be allocated every time the data node definition changes, or only sometimes?
  - (i) If only sometimes, then each SID needs to be map to (namespace, name, set of module revisions/versions), how would this mapping be maintained/updated?
  - (ii) If every time, then you have a similar mapping issue as per (i), only a single revision, but may also get a proliferation of SID values for the same logical path.

RW:

- B) Each YANG server only implements a single revision of a YANG module, and hence a single definition of a particular YANG data node.
  What happens if a server states that it's using module revision X, but then receives a SID that is bound to the leaf definition in a different module revision Y.
  Presumably the only thing that it can do is treat that as an unknown identifier/SID.
  Possibly it also opens up further possible issues between conflating the SID [breaks here]

— C) If you had a middlebox translating between SIDs and Names then it would also need to know the schema revision information to perform the translation correctly.

— [end of DISCUSS discussion from Robert Wilton] —

Opinion:

(C) is a bit of a killer for me.

But that proxy is hard to do for other reasons, so:
Can we reap the benefits of a static(*) SID-name mapping?

CRI

Concise Resource Identifier:
Concise equivalent of URIs and URI references (RFC 3986)

New representation format for **URI data model**

draft-ietf-core-href defines **CRIs** and CRI references

updates in -09

(-08: 2021-11-06, for 2021-11-08 IETF112 meeting)

Discussed at 2021-12-08:

- define CDDL features: no-authority, extended-cri (for percent-encoded text)
- percent-encoded text are byte strings; simplify!
  - also allow `pet` in fragment component

New in 2022-01-15 draft (-09):

— No longer remove lone empty path segment
  `s://foo ≠ s://foo/`

```
$ echo "cri'coap://foo/'" | diag2diag.rb -acri
[-1, ["foo"], [""]]
$ echo "cri'coap://foo'" | diag2diag.rb -acri
[-1, ["foo"]]
```

— Add an appendix "The small print"
  Collect the weird examples

Done: submit -09

Integrated CRI into app-EDN implementation in cbor-diag

Plan:

- Work more on implementations
- Get PR #12 (test vectors) ready for release

```
$ echo "cri'coap://foo'" | diag2pretty.rb -acri
82                  # array(2)
   20               # negative(0)
   81               # array(1)
      63            # text(3)
         666f6f     # "foo"
$ echo "cri'coap://foo/'" | diag2pretty.rb -acri
83                  # array(3)
   20               # negative(0)
   81               # array(1)
      63            # text(3)
         666f6f     # "foo"
   81               # array(1)
      60            # text(0)
                    # ""
```