IANA YANG SIDs Registry, Constrained YANG Library, Unified Datastore

CoRE Interim 07

Vojtěch Vilímek

2026-05-20

Agenda

- IANA YANG SIDs Registry
- Constrained YANG Library
- Unified Datastore Semantics
- PYCORECONF Discussion

IANA YANG SIDs Registry

Andy Bierman's comments

- no YANG Doctors review established for .sid Files (or YD experts review).
- iana-if-types: allocation too small

  SID range has 400 numbers, allocation for revision 2026-03-17 leaving 94 SIDs unassigned.
  Andy would like to have 10000 SID range.
  Could be solved by additional SID range when necessary.
- ietf-system, RFC9595 example is not correct RFC9595 .sid file example should not be considered normative.
- ietf-yang-types and ietf-inet-types: old reference to RFC6991 not RFC9911 – Fixed
- (NIT) ietf-coreconf entry has reference to RFC9595.

IANA YANG SIDs Registry

Link.

Errata wording then Registry population.

General: Should we use size 50 for typedef only modules?

File ietf-yang-types@2025-12-22.sid created
Number of SIDs available : 50
Number of SIDs used : 1

File ietf-inet-types@2025-12-22.sid created
Number of SIDs available : 40
Number of SIDs used : 1

IANA YANG SIDs Registry

Should we consider automatic output as good enough for standard .sid files?

Current modules in registry:
iana-crypt-hash, iana-if-types, ietf-coreconf, ietf-ient-types, ietf-interfaces, ietf-ip, ietf-netconf-acm, ietf-schc, ietf-sid-file, ietf-system, ietf-voucher-request, ietf-voucher, ietf-yang-types

Why ietf-sid-file? Only standardized encoding is JSON.

Constrained YANG Library

```
// file ietf-constrained-yang-library.yang
  list import-only-module {
    key "identifier revision";
    leaf revision {
      type union {
      type revision-identifier;
        type string; // would not be empty better?
          legnth 0;
        }
      }
      description
        "The YANG module revision date.";
    }
  }
```

Need to allocate SID range, and create .sid File.

Unified Datastore Semantics

2.4. Unified datastore

CORECONF supports a simple datastore model consisting of a single unified datastore. This datastore provides access to both configuration and operational data. Configuration updates performed on this datastore are reflected immediately or with a minimal delay as operational data.
More complex datastore models such as the Network Management Datastore Architecture (NMDA) as defined by [RFC8342] are out of scope of the present specification.

Unified Datastore Semantics

Characteristics of the unified datastore are summarized in the table below:

Name	**Value**
Name	unified
YANG modules	all modules
YANG nodes	all data nodes ("config true" and "config false")
Access	read-write
How applied	changes applied in place immediately or with a minimal delay
Protocols	CORECONF
Defined in	"ietf-coreconf"

Unified Datastore Semantics

Access	read-write

Are all data nodes read-write, or just the "config true"? The "config false" are read-only.

How applied	changes applied in place immediately or with a minimal delay

Do I have two value, where the "config true" shadows the "config false", or just one value?

Unified Datastore Semantics

Consistency

Operational data are always changing.
For more complex queries, data consistency may be broken during message processing.
Also for iPATCH requests, rollback functionality is requered for two-or-more elements of CBOR sequence.

Isn't the rollback too resource heavy for the constrained environment?

CORECONF Consistency

Examples

```
REQ: FETCH </c>  (Content-Format: application/yang-identifiers+cbor-seq)
[1533, "eth0"]   / interface (SID 1533) with name = "eth0" /
1723,            / current-datetime (SID 1723) /

RES: 2.05 Content  (Content-Format: application/yang-instances+cbor-seq)
{
  1533 : {
     4 : "eth0",              / name (SID 1537) /
    13 : "referenced-value"    / leafref to 1723 /
  }
}
/ in the meantime the instance 1723 caesed to exist /
{
  1723 : null          / current-datetime (SID 1723) /
},
```

CORECONF Consistency

Examples

```
REQ: iPATCH </c>   (Content-Format: application/yang-instances+cbor-seq)
{  1755 : true  }                / enabled (SID 1755) /
[  "obvious", "error" ]
{  1756 : {                      / server (SID 1756)  /
     3 : "tic.nrc.ca",           / name (SID 1759)    /
     4 : true,                   / prefer (SID 1760)  /
     5 : {                       / udp (SID 1761)     /
       1 : "132.246.11.231" }    / address (SID 1762) /
}  }

RES: 4.00 Bad Request
{ 1025: {
    4: 1019  / error-tag     (SID 1029) -> operation-failed  (SID 1019) /
    1: 1012  / error-app-tag (SID 1026) -> malformed-message (SID 1012) /
    3: "Expected map, got array"    / error-message (SID 1028) /
  }
}
```

PYCORECONF Discussion

- YANG-oblivious process
- all built-in types
- typedefs
- maybe work on sid-file-ext.yang?

PYCORECONF Discussion