Authors: M.V. Veillette, Ed.   A.P. Pelov, Ed.
         Trilliant Networks Inc.   Acklio
         I. Petrov, Ed.   C. Bormann
         Google Switzerland GmbH   Universität Bremen TZI
         M. Richardson
         Sandelman Software Works

YANG Schema Item iDentifier (YANG SID)

Abstract

   YANG Schema Item iDentifiers (YANG SID) are globally unique 63-bit
   unsigned integers used to identify YANG items, as a more compact
   method to identify YANG items that can be used for efficiency and in
   constrained environments (RFC 7228). This document defines the
   semantics, the registration, and assignment processes of YANG SIDs
   for IETF managed YANG modules. To enable the implementation of these
   processes, this document also defines a file format used to persist
   and publish assigned YANG SIDs.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Some of the items defined in YANG [RFC7950] require the use of a unique identifier. In both Network Configuration Protocol (NETCONF) [RFC6241] and RESTCONF [RFC8040], these identifiers are implemented using names. To allow the implementation of data models defined in YANG in constrained devices [RFC7228] and constrained networks, a more compact method to identify YANG items is required. This compact identifier, called YANG Schema Item iDentifier or YANG SID (or simply SID in this document and when the context is clear), is encoded using a 63-bit unsigned integer. The limitation to 63-bit unsigned integers allows SIDs to be manipulated more easily on platforms that might otherwise lack 64-bit unsigned arithmetic. The loss of a single bit of range is not significant given the size of the remaining space.

The following items are identified using SIDs:

  *identities

  *data nodes (Note: including those nodes defined by the 'yang-
   data' extension.)

  *remote procedure calls (RPCs) and associated input(s) and
   output(s)

  *actions and associated input(s) and output(s)

  *notifications and associated information

  *YANG modules and features

It is possible that some protocols use only a subset of the assigned SIDs, for example, for protocols equivalent to NETCONF [RFC6241] like [I-D.ietf-core-comi] the transportation of YANG module SIDs might be unnecessary. Other protocols might need to be able to transport this information, for example protocols related to discovery such as Constrained YANG Module Library [I-D.ietf-core-yang-library].

SIDs are globally unique integers. A registration system is used in order to guarantee their uniqueness. SIDs are registered in blocks called "SID ranges".

SIDs are assigned permanently. Items introduced by a new revision of a YANG module are added to the list of SIDs already assigned. Assignment of SIDs to YANG items are usually automated as discussed in Appendix B, which also discusses some cases where manual interventions may be appropriate.

Section 3 provides more details about the registration process of YANG modules and associated SIDs. To enable the implementation of this registry, Section 4 defines a standard file format used to store and publish SIDs.

IETF managed YANG modules that need to allocate SIDs use the IANA mechanism specified in this document. YANG modules created by other parties allocate SID ranges using the IANA allocation mechanisms via Mega-Ranges (see Section 7.3); within the Mega-Range allocation, those other parties are free to make up their own mechanism.

At the time of writing, a tool for automated ".sid" file generation is available as part of the open-source project PYANG [PYANG].

2. Terminology and Notation

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in [RFC7950]:

*action

*feature

*module

*notification

*RPC

*schema node

*schema tree

*submodule

The following term is defined in [RFC8040]:

*yang-data extension

This specification also makes use of the following terminology:

*item: A schema node, an identity, a module, or a feature defined using the YANG modeling language.

*schema-node path: A schema-node path is a string that identifies
 a schema node within the schema tree. A path consists of the list
 of consecutive schema node identifier(s) separated by slashes
 ("/"). Schema node identifier(s) are always listed from the top-
 level schema node up to the targeted schema node and could
 contain namespace information. (e.g. "/ietf-system:system-state/
 clock/current-datetime")

*Namespace-qualified form - a schema node identifier is prefixed
 with the name of the module in which the schema node is defined,
 separated from the schema node identifier by the colon character
 (":").

*YANG Schema Item iDentifier (YANG SID or simply SID): Unsigned
 integer used to identify different YANG items.

3. ".sid" file lifecycle

YANG is a language designed to model data accessed using one of the
compatible protocols (e.g. NETCONF [RFC6241], RESTCONF [RFC8040] and
CORECONF [I-D.ietf-core-comi]). A YANG module defines hierarchies of
data, including configuration, state data, RPCs, actions and
notifications.

Many YANG modules are not created in the context of constrained
applications. YANG modules can be implemented using NETCONF
[RFC6241] or RESTCONF [RFC8040] without the need to assign SIDs.

As needed, authors of YANG modules can assign SIDs to their YANG
modules. In order to do that, they should first obtain a SID range
from a registry and use that range to assign or generate SIDs to
items of their YANG module. The assignments can then be stored in a
".sid" file. For example on how this could be achieved, please refer
to Appendix C.

Registration of the ".sid" file associated to a YANG module is
optional but recommended to promote interoperability between devices
and to avoid duplicate allocation of SIDs to a single YANG module.
Different registries might have different requirements for the
registration and publication of the ".sid" files. For a diagram of
one of the possibilities, please refer to the activity diagram on
Figure 4 in Appendix C.

Each time a YANG module or one of its imported module(s) or included
sub-module(s) is updated, a new ".sid" file **MAY** be created if the
new or updated items will need SIDs. All the SIDs present in the
previous version of the ".sid" file **MUST** be present in the new
version as well. The creation of this new version of the ".sid" file
SHOULD be performed using an automated tool.

If a new revision requires more SIDs than initially allocated, a new
SID range **MUST** be added to the 'assignment-range' as defined in
Section 4. These extra SIDs are used for subsequent assignments.

For an example of this update process, see activity diagram Figure 5
in Appendix C.

4. ".sid" file format

".sid" files are used to persist and publish SIDs assigned to the
different YANG items of a specific YANG module. It has the following
structure.

```
module: ietf-sid-file

  structure sid-file:
    +-- module-name           yang:yang-identifier
    +-- module-revision?      revision-identifier
    +-- sid-file-version?     sid-file-version-identifier
    +-- description?          string
    +-- dependency-revision* [module-name]
    |  +-- module-name        yang:yang-identifier
    |  +-- module-revision    revision-identifier
    +-- assignment-range* [entry-point]
    |  +-- entry-point    sid
    |  +-- size           uint64
    +-- item* [namespace identifier]
       +-- namespace     enumeration
       +-- identifier    union
       +-- sid           sid
```

Figure 1: YANG tree for ietf-sid-file

The following YANG module defines the structure of this file,
encoding is performed in JSON [RFC8259] using the rules defined in
[RFC7951]. It references ietf-yang-types defined in [RFC6991] and
ietf-restconf defined in [RFC8040].

RFC Ed.: please update the date of the module and Copyright if
needed and remove this note.

```
<CODE BEGINS> file "ietf-sid-file@2021-11-16.yang"

module ietf-sid-file {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sid-file";
  prefix sid;

  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types.";
  }
  import ietf-yang-structure-ext {
    prefix sx;
    reference "RFC 8791: YANG Data Structure Extensions.";
  }

  organization
    "IETF Core Working Group";

  contact
    "WG Web:    <https://datatracker.ietf.org/wg/core/>

     WG List:  <mailto:core@ietf.org>

     Editor:   Michel Veillette
               <mailto:michel.veillette@trilliant.com>

     Editor:   Andy Bierman
               <mailto:andy@yumaworks.com>

     Editor:   Alexander Pelov
               <mailto:a@ackl.io>

     Editor:   Ivaylo Petrov
               <mailto:ivaylopetrov@google.com>";

  description
    "Copyright (c) 2021 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.
```

```
   The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
   NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
   'MAY', and 'OPTIONAL' in this document are to be interpreted as
   described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
   they appear in all capitals, as shown here.

   This module defines the structure of the .sid files.

   Each .sid file contains the mapping between each
   string identifier defined by a YANG module and a
   corresponding numeric value called YANG SID.";

revision 2021-11-16 {
  description
    "Initial revision.";
  reference
    "[RFC XXXX] YANG Schema Item iDentifier (YANG SID)";
}

typedef revision-identifier {
  type string {
    pattern '\d{4}-\d{2}-\d{2}';
  }
  description
    "Represents a date in YYYY-MM-DD format.";
}

typedef sid-file-version-identifier {
  type uint32;
  description
    "Represents the version of a .sid file.";
}

typedef sid {
  type uint64 {
    range "0..9223372036854775807";
  }
  description
    "YANG Schema Item iDentifier";
  reference
    "[RFC XXXX] YANG Schema Item iDentifier (YANG SID)";
}

typedef schema-node-path {
  type string {
    pattern
      '/[a-zA-Z_][a-zA-Z0-9\-_.]*:[a-zA-Z_][a-zA-Z0-9\-_.]*' +
      '(/[a-zA-Z_][a-zA-Z0-9\-_.]*(:[a-zA-Z_][a-zA-Z0-9\-_.]*)?)*';
  }
```

```
    description
      "A schema-node path is an absolute YANG schema node identifier
      as defined by the YANG ABNF rule absolute-schema-nodeid,
      except that module names are used instead of prefixes.

      This string additionally follows the following rules:

        o  The leftmost (top-level) data node name is always in the
           namespace-qualified form.
        o  Any subsequent schema node name is in the
           namespace-qualified form if the node is defined in a module
           other than its parent node, and the simple form is used
           otherwise. No predicates are allowed.";
    reference
      "RFC 7950, The YANG 1.1 Data Modeling Language;
       Section 6.5: Schema Node Identifier;";
}

sx:structure sid-file {
    uses sid-file-contents;
}

grouping sid-file {
  description "A grouping that contains a YANG container
    representing the file structure of a .sid files.";

  container sid-file {
    description
      "A Wrapper container that together with the rc:yang-data
       extension marks the YANG data structures inside as not being
       intended to be implemented as part of a configuration
       datastore or as an operational state within the server.";
    uses sid-file-contents;
  }
}

grouping sid-file-contents {
  description
    "A grouping that defines the contents of a container that
     represente the file structure of a .sid files.";

  leaf module-name {
    type yang:yang-identifier;
    mandatory true;
    description
      "Name of the YANG module associated with this .sid file.";
  }

  leaf module-revision {
    type revision-identifier;
```

```
    description
      "Revision of the YANG module associated with this .sid
      file.
      This leaf is not present if no revision statement is
      defined in the YANG module.";
}

leaf sid-file-version {
  type sid-file-version-identifier;
  default 0;
  description
    "Optional leaf that specifies the version number of the
    .sid file.  .sid files and the version sequence are
    specific to a given YANG module revision. This number
    starts at zero when there is a new YANG module revision and
    increases monotonically.  This number can distinguish
    updates to the .sid file which are the result of new
    processing, or the result of reported errata.";
}

leaf description {
  type string;
  description
    "Free-form meta information about the generated file. It
    might include .sid file generation tool and time among
    other things.";
}

list dependency-revision {
  key "module-name";

  description
    "Information about the used revision during the .sid file
    generation of each YANG module that the module in
    'module-name' imported.";

  leaf module-name {
    type yang:yang-identifier;
    description
      "Name of the YANG module, dependency of 'module-name',
      for which revision information is provided.";
  }
  leaf module-revision {
    type revision-identifier;
    mandatory true;
    description
      "Revision of the YANG module, dependency of
      'module-name', for which revision information is
      provided.";
```

```
    }
  }

  list assignment-range {
    key "entry-point";
    description
      "YANG SID range(s) allocated to the YANG module identified
      by 'module-name' and 'module-revision'.

      - The YANG SID range first available value is entry-point
        and the last available value in the range is
        (entry-point + size - 1).
      - The YANG SID ranges specified by all assignment-rages
        MUST NOT overlap.";

    leaf entry-point {
      type sid;
      description
        "Lowest YANG SID available for assignment.";
    }

    leaf size {
      type uint64;
      mandatory true;
      description
        "Number of YANG SIDs available for assignment.";
    }
  }

  list item {
    key "namespace identifier";
    unique "sid";

    description
      "Each entry within this list defined the mapping between
      a YANG item string identifier and a YANG SID. This list
      MUST include a mapping entry for each YANG item defined by
      the YANG module identified by 'module-name' and
      'module-revision'.";

    leaf namespace {
      type enumeration {
        enum module {
          value 0;
          description
            "All module and submodule names share the same
            global module identifier namespace.";
        }
        enum identity {
```

```
        value 1;
        description
          "All identity names defined in a module and its
          submodules share the same identity identifier
          namespace.";
      }
      enum feature {
        value 2;
        description
          "All feature names defined in a module and its
          submodules share the same feature identifier
          namespace.";
      }
      enum data {
        value 3;
        description
          "The namespace for all data nodes, as defined in
          YANG.";
      }
    }
    description
      "Namespace of the YANG item for this mapping entry.";
  }

  leaf identifier {
    type union {
      type yang:yang-identifier;
      type schema-node-path;
    }
    description
      "String identifier of the YANG item for this mapping
      entry.

      If the corresponding 'namespace' field is 'module',
      'feature', or 'identity', then this field MUST
      contain a valid YANG identifier string.

      If the corresponding 'namespace' field is 'data',
      then this field MUST contain a valid schema node
      path.";
  }

  leaf sid {
    type sid;
    mandatory true;
    description
      "YANG SID assigned to the YANG item for this mapping
      entry.";
  }
```

```
        }
      }
}

<CODE ENDS>
```

                    Figure 2: YANG module ietf-sid-file

5. Content-Types

   The following Content-Type has been defined in [I-D.ietf-core-yang-cbor]:

   application/yang-data+cbor; id=sid: This Content-Type represents a
      CBOR YANG document containing one or multiple data node values.
      Each data node is identified by its associated SID.

      FORMAT: CBOR map of SID, instance-value

      The message payload of Content-Type 'application/yang-data+cbor'
      is encoded using a CBOR map. Each entry within the CBOR map
      contains the data node identifier (i.e. SID) and the associated
      instance-value. Instance-values are encoded using the rules
      defined in Section 4 of [I-D.ietf-core-yang-cbor].

6. Security Considerations

   This document defines a new type of identifier used to encode data
   that are modeled in YANG [RFC7950]. This new identifier maps
   semantic concepts to integers, and if the source of this mapping is
   not trusted, then new security risks might occur if an attacker can
   control the mapping.

   At the time of writing, it is expected that the SID files will be
   processed by a software developer, within a software development
   environment. Developers are advised to only import SID files from
   authoritative sources. IANA is the authoritative source for IETF
   managed YANG modules.

   Conceptually, SID files could be processed by less-constrained
   target systems such as network management systems. Such systems need
   to take extra care to make sure that they are only processing SID
   files from authoritative sources, as authoritative as the YANG
   modules that they are using.

7. IANA Considerations

7.1. YANG Namespace Registration

This document registers the following XML namespace URN in the "IETF
XML Registry", following the format defined in [RFC3688]:

URI: please assign urn:ietf:params:xml:ns:yang:ietf-sid-file

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

Reference: RFC XXXX

// RFC Ed.: please replace XXXX with RFC number and remove this note

7.2. Register ".sid" File Format Module

This document registers one YANG module in the "YANG Module Names"
registry [RFC6020]:

  *name: ietf-sid-file

  *namespace: urn:ietf:params:xml:ns:yang:ietf-sid-file

  *prefix: sid

  *reference: RFC XXXX

// RFC Ed.: please replace XXXX with RFC number and remove this note

7.3. Create new IANA Registry: "YANG SID Mega-Range" registry

The name of this registry is "YANG SID Mega-Range". This registry is
used to record the delegation of the management of a block of SIDs
to third parties (such as SDOs or registrars).

7.3.1. Structure

Each entry in this registry must include:

  *The entry point (first SID) of the registered SID block.

  *The size of the registered SID block. The size **SHOULD** be one
   million (1 000 000) SIDs, it **MAY** exceptionally be a multiple of
   1 000 000.

  *The contact information of the requesting organization including:

      -The policy of SID range allocations: Public, Private or Both.

-Organization name

-URL

7.3.2. Allocation policy

The IANA policy for future additions to this registry is "Expert Review" [RFC8126].

An organization requesting to manage a YANG SID Range (and thus have an entry in the YANG SID Mega-Range Registry), must ensure the following capacities:

  *The capacity to manage and operate a YANG SID Range Registry. A YANG SID Range Registry **MUST** provide the following information for all YANG SID Ranges allocated by the Registry:

    -Entry Point of allocated YANG SID Range

    -Size of allocated YANG SID Range

    -Type: Public or Private

       oPublic Ranges **MUST** include at least a reference to the YANG module and ".sid" files for that YANG SID Range (e.g., compare Section 7.4.3 for the IETF YANG SID registry).

       oPrivate Ranges **MUST** be marked as "Private"

  *A Policy of allocation, which clearly identifies if the YANG SID Range allocations would be Private, Public or Both.

  *Technical capacity to ensure the sustained operation of the registry for a period of at least 5 years. If Private Registrations are allowed, the period must be of at least 10 years.

If a size of the allocation beyond 1 000 000 is desired, the organization must demonstrate the sustainability of the technical approach for utilizing this size of allocation and how it does not negatively impact the overall usability of the SID allocation mechanisms; such allocations are preferably placed in the space above 4 295 000 000 (64-bit space).

7.3.2.1. First allocation

For a first allocation to be provided, the requesting organization must demonstrate a functional registry infrastructure.

7.3.2.2. Consecutive allocations

On subsequent allocation request(s), the organization must
demonstrate the exhaustion of the prior range. These conditions need
to be asserted by the assigned expert(s).

If that extra-allocation is done within 3 years from the last
allocation, the experts need to discuss this request on the CORE
working group mailing list and consensus needs to be obtained before
allocating a new Mega-Range.

7.3.3. Initial contents of the Registry

The initial entry in this registry is allocated to IANA:

Entry Point	Size	Allocation	Organization name	URL
0	1000000	Public	IANA	iana.org

Table 1

7.4. Create a new IANA Registry: IETF YANG SID Range Registry (managed by IANA)

7.4.1. Structure

Each entry in this registry must include:

  *The SID range entry point.

  *The SID range size.

  *The YANG module name.

  *Document reference.

7.4.2. Allocation policy

The first million SIDs assigned to IANA is sub-divided as follows:

  *The range of 0 to 999 (size 1000) is subject to "IESG Approval"
   as defined in [RFC8126]; of these, SID value 0 has been reserved
   for implementations to internally signify the absence of a SID
   number and does not occur in interchange.

  *The range of 1000 to 59,999 (size 59,000) is designated for YANG
   modules defined in RFCs.

    -The IANA policy for additions to this registry is either:

       o"Expert Review" [RFC8126] in case the ".sid" file comes
        from a YANG module from an existing RFC, or

o"RFC Required" [RFC8126] otherwise.

-The Expert **MUST** verify that the YANG module for which this
 allocation is made has an RFC (existing RFC) OR is on track to
 become RFC (early allocation with a request from the WG chairs
 as defined by [BCP100]).

*The range of 60,000 to 99,999 (size 40,000) is reserved for
 experimental YANG modules. This range **MUST NOT** be used in
 operational deployments since these SIDs are not globally unique
 which limit their interoperability. The IANA policy for this
 range is "Experimental use" [RFC8126].

*The range of 100,000 to 999,999 (size 900,000) is "Reserved" as
 defined in [RFC8126].

Entry Point	Size	IANA policy
0	1,000	IESG Approval
1,000	59,000	RFC Required
60,000	40,000	Experimental/Private use
100,000	900,000	Reserved

Table 2

The size of the SID range allocated for a YANG module is recommended
to be a multiple of 50 and to be at least 33% above the current
number of YANG items. This headroom allows assignment within the
same range of new YANG items introduced by subsequent revisions. The
SID range size **SHOULD NOT** exceed 1000; a larger size may be
requested by the authors if this recommendation is considered
insufficient. It is important to note that an additional SID range
can be allocated to an existing YANG module if the initial range is
exhausted; this then just leads to slightly less efficient
representation.

In case a SID range is allocated for an existing RFC through the
"Expert Review" policy, the Document reference field for the given
allocation should point to the RFC that the YANG module is defined
in.

In case a SID range is required before publishing the RFC due to
implementations needing stable SID values, early allocation as
defined in [BCP100] can be employed. As specified in Section 4.6 of
[RFC8126], RFCs and by extension documents that are expected to
become an RFC fulfill the requirement for "Specification Required"
stated in Section 2 of [BCP100], which allows for the early
allocation process to be employed.

7.4.3. Publication of the ".sid" file

For a YANG module approved for publication as an RFC, a ".sid" file
SHOULD be included in the Internet-Draft as a source code block.
This ".sid" file is to be extracted by IANA/the expert reviewer and
put into the YANG SID Registry (Section 7.5) along with the YANG
module. The ".sid" file **MUST NOT** be published as part of the RFC:
the IANA Registry is authoritative and a link is to be inserted in
the RFC.

7.4.4. Initial contents of the registry

Initial entries in this registry are as follows:

Entry Point	Size	Module name	Document reference
0	1	(Reserved: not a valid SID)	RFCXXXX
1000	100	ietf-coreconf	[I-D.ietf-core-comi]
1100	50	ietf-yang-types	[RFC6991]
1150	50	ietf-inet-types	[RFC6991]
1200	50	iana-crypt-hash	[RFC7317]
1250	50	ietf-netconf-acm	[RFC8341]
1300	50	ietf-sid-file	RFCXXXX
1500	100	ietf-interfaces	[RFC8343]
1600	100	ietf-ip	[RFC8344]
1700	100	ietf-system	[RFC7317]
1800	400	iana-if-type	[RFC7224]
2400	50	ietf-voucher	[RFC8366]
2450	50	ietf-constrained-voucher	[I-D.ietf-anima-constrained-voucher]
2500	50	ietf-constrained-voucher-request	[I-D.ietf-anima-constrained-voucher]

Table 3

// RFC Ed.: replace XXXX with RFC number assigned to this draft.

For allocation, RFC publication of the YANG module is required as
per [RFC8126]. The YANG module must be registered in the "YANG
module Name" registry according to the rules specified in Section 14
of [RFC6020].

7.5. Create new IANA Registry: "IETF YANG SID Registry"

The name of this registry is "IETF YANG SID Registry". This registry
is used to record the allocation of SIDs for individual YANG module
items.

7.5.1. Structure

Each entry in this registry must include:

  *The YANG module name. This module name must be present in the
   "Name" column of the "YANG Module Names" registry.

  *A link to the associated ".yang" file. This file link must be
   present in the "File" column of the "YANG Module Names" registry.

  *The link to the ".sid" file which defines the allocation. The
   ".sid" file is stored by IANA.

  *The number of actually allocated SIDs in the ".sid" file.

7.5.2. Allocation policy

The allocation policy is Expert review. The Expert **MUST** ensure that
the following conditions are met:

  *The ".sid" file has a valid structure:

    -The ".sid" file **MUST** be a valid JSON file following the
     structure of the module defined in RFCXXXX (RFC Ed: replace
     XXX with RFC number assigned to this draft).

  *The ".sid" file allocates individual SIDs ONLY in the YANG SID
   Ranges for this YANG module (as allocated in the IETF YANG SID
   Range Registry):

    -All SIDs in this ".sid" file **MUST** be within the ranges
     allocated to this YANG module in the "IETF YANG SID Range
     Registry".

  *If another ".sid" file has already allocated SIDs for this YANG
   module (e.g. for older or newer versions of the YANG module), the
   YANG items are assigned the same SIDs as in the other ".sid"
   file.

  *If there is an older version of the ".sid" file, all allocated
   SIDs from that version are still present in the current version
   of the ".sid" file.

7.5.3. Recursive Allocation of YANG SID Range at Document Adoption

Due to the difficulty in changing SID values during IETF document
processing, it is expected that most documents will ask for SID
allocations using Early Allocations [BCP100]. The details of the
Early Allocation should be included in any Working Group Adoption
call. Prior to Working Group Adoption, an internet draft author can

use the experimental SID range (as per [Section 7.4.2](#)) for their SIDs allocations or other values that do not create ambiguity with other SID uses (for example they can use a range that comes from a non-IANA managed "YANG SID Mega-Range" registry).

After Working Group Adoption, any modification of a ".sid" file is expected to be discussed on the mailing list of the appropriate Working Groups. Specific attention should be paid to implementers' opinion after Working Group Last Call if a SID value is to change its meaning. In all cases, a ".sid" file and the SIDs associated with it are subject to change before the publication of an internet draft as an RFC.

During the early use of SIDs, many existing, previously published YANG modules will not have SID allocations. For an allocation to be useful the included YANG modules may also need to have SID allocations made.

The Expert Reviewer who performs the (Early) Allocation analysis will need to go through the list of included YANG modules and perform SID allocations for those modules as well.

  *If the document is a published RFC, then the allocation of SIDs for its referenced YANG modules is permanent. The Expert Reviewer provides the generated ".sid" file to IANA for registration. This process may be time-consuming during a bootstrap period (there are over 100 YANG modules to date, none of which have SID allocations), but should quiet down once needed entries are allocated.

  *If the document is an unprocessed Internet-Draft adopted in a WG, then an Early Allocation is performed for this document as well. Early Allocations require approval by an IESG Area Director. An early allocation which requires additional allocations will list the other allocations in its description, and will be cross-posted to the any other working group mailing lists.

  *A YANG module which references a module in a document which has not yet been adopted by any working group will be unable to perform an Early Allocation for that other document until it is adopted by a working group. As described in [[BCP100](#)], an AD Sponsored document acts as if it had a working group. The approving AD may also exempt a document from this policy by agreeing to AD Sponsor the document.

At the end of the IETF process all the dependencies of a given module for which SIDs are assigned, should also have SIDs assigned. Those dependencies' assignments should be permanent (not Early Allocation).

A previously SID-allocated YANG module which changes its references to include a YANG module for which there is no SID allocation needs to repeat the Early Allocation process.

Early Allocations are made with a one-year period, after which they are expired. [BCP100] indicates that at most one renewal may be made. For the SID allocation a far more lenient stance is desired.

1. An extension of a referencing documents Early Allocation should update any referenced Early Allocations to expire no sooner than the referencing document.

2. The [BCP100] mechanism allows the IESG to provide a second renewal, and such an event may prompt some thought about how the collection of documents are being processed.

This is driven by the very generous size of the SID space and the often complex and deep dependencies of YANG modules. Often a core module with many dependencies will undergo extensive review, delaying the publication of other documents.

[BCP100] also says:

Note that if a document is submitted for review to the IESG and at the time of submission some early allocations are valid (not expired), these allocations should not be expired while the document is under IESG consideration or waiting in the RFC Editor's queue after approval by the IESG.

7.5.4. Initial contents of the registry

None.

8. References

8.1. Normative References

[BCP100]   Cotton, M., "Early IANA Allocation of Standards Track
            Code Points", BCP 100, RFC 7120, January 2014.
            <https://www.rfc-editor.org/info/bcp100>

[I-D.ietf-core-yang-cbor] Veillette, M., Petrov, I., Pelov, A.,
            Bormann, C., and M. Richardson, "CBOR Encoding of Data
            Modeled with YANG", Work in Progress, Internet-Draft,
            draft-ietf-core-yang-cbor-17, 25 October 2021, <https://
            www.ietf.org/archive/id/draft-ietf-core-yang-
            cbor-17.txt>.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/

RFC2119, March 1997, <https://www.rfc-editor.org/info/
rfc2119>.

[RFC3688]   Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
            DOI 10.17487/RFC3688, January 2004, <https://www.rfc-
            editor.org/info/rfc3688>.

[RFC6991]   Schoenwaelder, J., Ed., "Common YANG Data Types", RFC
            6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-
            editor.org/info/rfc6991>.

[RFC7950]   Bjorklund, M., Ed., "The YANG 1.1 Data Modeling
            Language", RFC 7950, DOI 10.17487/RFC7950, August 2016,
            <https://www.rfc-editor.org/info/rfc7950>.

[RFC7951]   Lhotka, L., "JSON Encoding of Data Modeled with YANG",
            RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://
            www.rfc-editor.org/info/rfc7951>.

[RFC8040]   Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
            Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,
            <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174]   Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
            2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
            May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259]   Bray, T., Ed., "The JavaScript Object Notation (JSON)
            Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/
            RFC8259, December 2017, <https://www.rfc-editor.org/info/
            rfc8259>.

8.2. Informative References

[I-D.ietf-anima-constrained-voucher] Richardson, M., Stok, P. V. D.,
            Kampanakis, P., and E. Dijk, "Constrained Bootstrapping
            Remote Secure Key Infrastructure (BRSKI)", Work in
            Progress, Internet-Draft, draft-ietf-anima-constrained-
            voucher-14, 25 October 2021, <https://www.ietf.org/
            archive/id/draft-ietf-anima-constrained-voucher-14.txt>.

[I-D.ietf-core-comi] Veillette, M., Stok, P. V. D., Pelov, A.,
            Bierman, A., and I. Petrov, "CoAP Management Interface
            (CORECONF)", Work in Progress, Internet-Draft, draft-
            ietf-core-comi-11, 17 January 2021, <https://
            www.ietf.org/archive/id/draft-ietf-core-comi-11.txt>.

[I-D.ietf-core-yang-library] Veillette, M. and I. Petrov,
            "Constrained YANG Module Library", Work in Progress,
            Internet-Draft, draft-ietf-core-yang-library-03, 11

January 2021, <https://www.ietf.org/archive/id/draft-ietf-core-yang-library-03.txt>.

[PYANG]   Bjorklund, M., "An extensible YANG validator and
            converter in python", <https://github.com/mbj4668/pyang>.

[RFC6020]   Bjorklund, M., Ed., "YANG - A Data Modeling Language for
            the Network Configuration Protocol (NETCONF)", RFC 6020,
            DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241]   Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J.,
            Ed., and A. Bierman, Ed., "Network Configuration Protocol
            (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
            <https://www.rfc-editor.org/info/rfc6241>.

[RFC7224]   Bjorklund, M., "IANA Interface Type YANG Module", RFC
            7224, DOI 10.17487/RFC7224, May 2014, <https://www.rfc-editor.org/info/rfc7224>.

[RFC7228]   Bormann, C., Ersue, M., and A. Keranen, "Terminology for
            Constrained-Node Networks", RFC 7228, DOI 10.17487/
            RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC7317]   Bierman, A. and M. Bjorklund, "A YANG Data Model for
            System Management", RFC 7317, DOI 10.17487/RFC7317,
            August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC8126]   Cotton, M., Leiba, B., and T. Narten, "Guidelines for
            Writing an IANA Considerations Section in RFCs", BCP 26,
            RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8341]   Bierman, A. and M. Bjorklund, "Network Configuration
            Access Control Model", STD 91, RFC 8341, DOI 10.17487/
            RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8343]   Bjorklund, M., "A YANG Data Model for Interface
            Management", RFC 8343, DOI 10.17487/RFC8343, March 2018,
            <https://www.rfc-editor.org/info/rfc8343>.

[RFC8344]   Bjorklund, M., "A YANG Data Model for IP Management", RFC
            8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

[RFC8366]   Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,
            "A Voucher Artifact for Bootstrapping Protocols", RFC

                     8366, DOI 10.17487/RFC8366, May 2018, <https://www.rfc-
                     editor.org/info/rfc8366>.

Appendix A. ".sid" file example

   The following ".sid" file (ietf-system@2014-08-06.sid) has been
   generated using the following yang modules:

     *ietf-system@2014-08-06.yang (defined in [RFC7317])

     *ietf-yang-types@2013-07-15.yang (defined in [RFC6991])

     *ietf-inet-types@2013-07-15.yang (defined in [RFC6991])

     *ietf-netconf-acm@2018-02-14.yang (defined in [RFC8341])

     *iana-crypt-hash@2014-08-06.yang (defined in [RFC7317])

   For purposes of exposition, line breaks have been introduced below
   in some JSON strings that represent overly long identifiers.

```json
{
  "ietf-sid-file:sid-file" : {
    "module-name": "ietf-system",
    "module-revision": "2014-08-06",
    "dependency-revision": [
      {
        "module-name": "ietf-yang-types",
        "module-revision": "2013-07-15"
      },
      {
        "module-name": "ietf-inet-types",
        "module-revision": "2013-07-15"
      },
      {
        "module-name": "ietf-netconf-acm",
        "module-revision": "2018-02-14"
      },
      {
        "module-name": "iana-crypt-hash",
        "module-revision": "2014-08-06"
      }
    ],
    "description": "Example sid file",
    "assignment-range": [
      {
        "entry-point": 1700,
        "size": 100
      }
    ],
    "item": [
      {
        "namespace": "module",
        "identifier": "ietf-system",
        "sid": 1700
      },
      {
        "namespace": "identity",
        "identifier": "authentication-method",
        "sid": 1701
      },
      {
        "namespace": "identity",
        "identifier": "local-users",
        "sid": 1702
      },
      {
        "namespace": "identity",
        "identifier": "radius",
        "sid": 1703
```

    },
    {
      "namespace": "identity",
      "identifier": "radius-authentication-type",
      "sid": 1704
    },
    {
      "namespace": "identity",
      "identifier": "radius-chap",
      "sid": 1705
    },
    {
      "namespace": "identity",
      "identifier": "radius-pap",
      "sid": 1706
    },
    {
      "namespace": "feature",
      "identifier": "authentication",
      "sid": 1707
    },
    {
      "namespace": "feature",
      "identifier": "dns-udp-tcp-port",
      "sid": 1708
    },
    {
      "namespace": "feature",
      "identifier": "local-users",
      "sid": 1709
    },
    {
      "namespace": "feature",
      "identifier": "ntp",
      "sid": 1710
    },
    {
      "namespace": "feature",
      "identifier": "ntp-udp-port",
      "sid": 1711
    },
    {
      "namespace": "feature",
      "identifier": "radius",
      "sid": 1712
    },
    {
      "namespace": "feature",
      "identifier": "radius-authentication",

```json
      "sid": 1713
    },
    {
      "namespace": "feature",
      "identifier": "timezone-name",
      "sid": 1714
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:set-current-datetime",
      "sid": 1715
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:set-current-datetime/
                    current-datetime",
      "sid": 1716
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system",
      "sid": 1717
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system-restart",
      "sid": 1718
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system-shutdown",
      "sid": 1719
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system-state",
      "sid": 1720
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system-state/clock",
      "sid": 1721
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system-state/clock/boot-datetime",
      "sid": 1722
    },
    {
```

```
    "namespace": "data",
    "identifier": "/ietf-system:system-state/clock/
                   current-datetime",
    "sid": 1723
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system-state/platform",
    "sid": 1724
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system-state/platform/machine",
    "sid": 1725
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system-state/platform/os-name",
    "sid": 1726
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system-state/platform/os-release",
    "sid": 1727
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system-state/platform/os-version",
    "sid": 1728
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/authentication",
    "sid": 1729
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/authentication/user",
    "sid": 1730
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/authentication/
                   user-authentication-order",
    "sid": 1731
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/authentication/user/
```

```
                  authorized-key",
      "sid": 1732
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/authentication/user/
                  authorized-key/algorithm",
      "sid": 1733
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/authentication/user/
                  authorized-key/key-data",
      "sid": 1734
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/authentication/user/
                  authorized-key/name",
      "sid": 1735
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/authentication/user/
                  name",
      "sid": 1736
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/authentication/user/
                  password",
      "sid": 1737
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/clock",
      "sid": 1738
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/clock/timezone-name",
      "sid": 1739
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/clock/timezone-utc-offset",
      "sid": 1740
    },
    {
```

```json
    "namespace": "data",
    "identifier": "/ietf-system:system/contact",
    "sid": 1741
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/dns-resolver",
    "sid": 1742
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/dns-resolver/options",
    "sid": 1743
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/dns-resolver/options/
                attempts",
    "sid": 1744
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/dns-resolver/options/
                timeout",
    "sid": 1745
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/dns-resolver/search",
    "sid": 1746
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/dns-resolver/server",
    "sid": 1747
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/dns-resolver/server/name",
    "sid": 1748
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/dns-resolver/server/
                udp-and-tcp",
    "sid": 1749
  },
  {
    "namespace": "data",
```

```json
    "identifier": "/ietf-system:system/dns-resolver/server/
                   udp-and-tcp/address",
    "sid": 1750
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/dns-resolver/server/
                   udp-and-tcp/port",
    "sid": 1751
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/hostname",
    "sid": 1752
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/location",
    "sid": 1753
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/ntp",
    "sid": 1754
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/ntp/enabled",
    "sid": 1755
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/ntp/server",
    "sid": 1756
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/ntp/server/
                   association-type",
    "sid": 1757
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/ntp/server/iburst",
    "sid": 1758
  },
  {
    "namespace": "data",
    "identifier": "/ietf-system:system/ntp/server/name",
```

```json
      "sid": 1759
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/ntp/server/prefer",
      "sid": 1760
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/ntp/server/udp",
      "sid": 1761
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/ntp/server/udp/address",
      "sid": 1762
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/ntp/server/udp/port",
      "sid": 1763
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/radius",
      "sid": 1764
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/radius/options",
      "sid": 1765
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/radius/options/attempts",
      "sid": 1766
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/radius/options/timeout",
      "sid": 1767
    },
    {
      "namespace": "data",
      "identifier": "/ietf-system:system/radius/server",
      "sid": 1768
    },
    {
      "namespace": "data",
```

```
          "identifier": "/ietf-system:system/radius/server/
                      authentication-type",
          "sid": 1769
        },
        {
          "namespace": "data",
          "identifier": "/ietf-system:system/radius/server/name",
          "sid": 1770
        },
        {
          "namespace": "data",
          "identifier": "/ietf-system:system/radius/server/udp",
          "sid": 1771
        },
        {
          "namespace": "data",
          "identifier": "/ietf-system:system/radius/server/udp/
                      address",
          "sid": 1772
        },
        {
          "namespace": "data",
          "identifier": "/ietf-system:system/radius/server/udp/
                      authentication-port",
          "sid": 1773
        },
        {
          "namespace": "data",
          "identifier": "/ietf-system:system/radius/server/udp/
                      shared-secret",
          "sid": 1774
        }
      ]
    }
}
```

   Figure 3: Example .sid file (ietf-system, with extra line-breaks)

   For reconstructing the actual JSON file from this figure, all line
   breaks that occur in what would be JSON strings need to be removed,
   including any following blank space (indentation) on the line after
   the line break; in each such case, a single identifier without any
   embedded blank space results. This removal can be accomplished with
   this simple Ruby script:

```
@u = %{[^"\n]*}; @q = @u + '"'
puts ARGF.read.gsub(/^(#@q(#@q#@q)*#@u) *\n +(#@q)/, "\\1\\3")
```

Appendix B. SID auto generation

Assignment of SIDs to YANG items **SHOULD** be automated. The recommended process to assign SIDs is as follows:

1. A tool extracts the different items defined for a specific YANG module.

2. The list of items is sorted in alphabetical order, 'namespace' in descending order, 'identifier' in ascending order. The 'namespace' and 'identifier' formats are described in the YANG module 'ietf-sid-file' defined in [Section 4](#).

3. SIDs are assigned sequentially from the entry point up to the size of the registered SID range. This approach is recommended to minimize the serialization overhead, especially when delta between a reference SID and the current SID is used by protocols aiming to reduce message size.

4. If the number of items exceeds the SID range(s) allocated to a YANG module, an extra range is added for subsequent assignments.

5. The "dependency-revision" should reflect the revision numbers of each YANG module that the YANG module imports at the moment of the generation.

When updating a YANG module that is in active use, the existing SID assignments are maintained. (In contrast, when evolving an early draft that has not yet been adopted by a community of developers, SID assignments are often better done from scratch after a revision.) If the name of a schema node changes, but the data remain structurally and semantically similar to what was previously available under an old name, the SID that was used for the old name **MAY** continue to be used for the new name. If the meaning of an item changes, a new SID **MAY** be assigned to it; this is particularly useful to allow the new SID to identify the new structure or semantics of the item. If the YANG data type changes in a new revision of a published module, such that the resulting CBOR encoding is changed, then implementations will be aided significantly if a new SID is assigned. Note that these decisions are generally at the discretion of the YANG module author, who should decide if the benefits of a manual intervention are worth the deviation from automatic assignment.

In case of an update to an existing ".sid" file, an additional step is needed that increments the ".sid" file version number. If there was no version number in the previous version of the ".sid" file, 0 is assumed as the version number of the old version of the ".sid" file and the version number is 1 for the new ".sid" file. Apart from

that, changes of ".sid" files can also be automated using the same method described above, only unassigned YANG items are processed at step #3. Already existing items in the ".sid" file should not be given new SIDs.

Note that ".sid" file versions are specific to a YANG module revision. For each new YANG module or each new revision of an existing YANG module, the version number of the initial ".sid" file should either be 0 or should not be present.

Note also that RPC or action "input" and "output" data nodes **MUST** always be assigned SID even if they don't contain data nodes. The reason for this requirement is that other modules can augment the given module and those SIDs might be necessary.

Appendix C. ".sid" file lifecycle

Before assigning SIDs to their YANG modules, YANG module authors must acquire a SID range from a "YANG SID Range Registry". If the YANG module is part of an IETF draft or RFC, the SID range need to be acquired from the "IETF YANG SID Range Registry" as defined in Section 7.4. For the other YANG modules, the authors can acquire a SID range from any "YANG SID Range Registry" of their choice.

Once the SID range is acquired, owners can use it to generate ".sid" file/s for their YANG module/s. It is recommended to leave some unallocated SIDs following the allocated range in each ".sid" file in order to allow better evolution of the YANG module in the future. Generation of ".sid" files should be performed using an automated tool. Note that ".sid" files can only be generated for YANG modules and not for submodules.

C.1. ".sid" File Creation

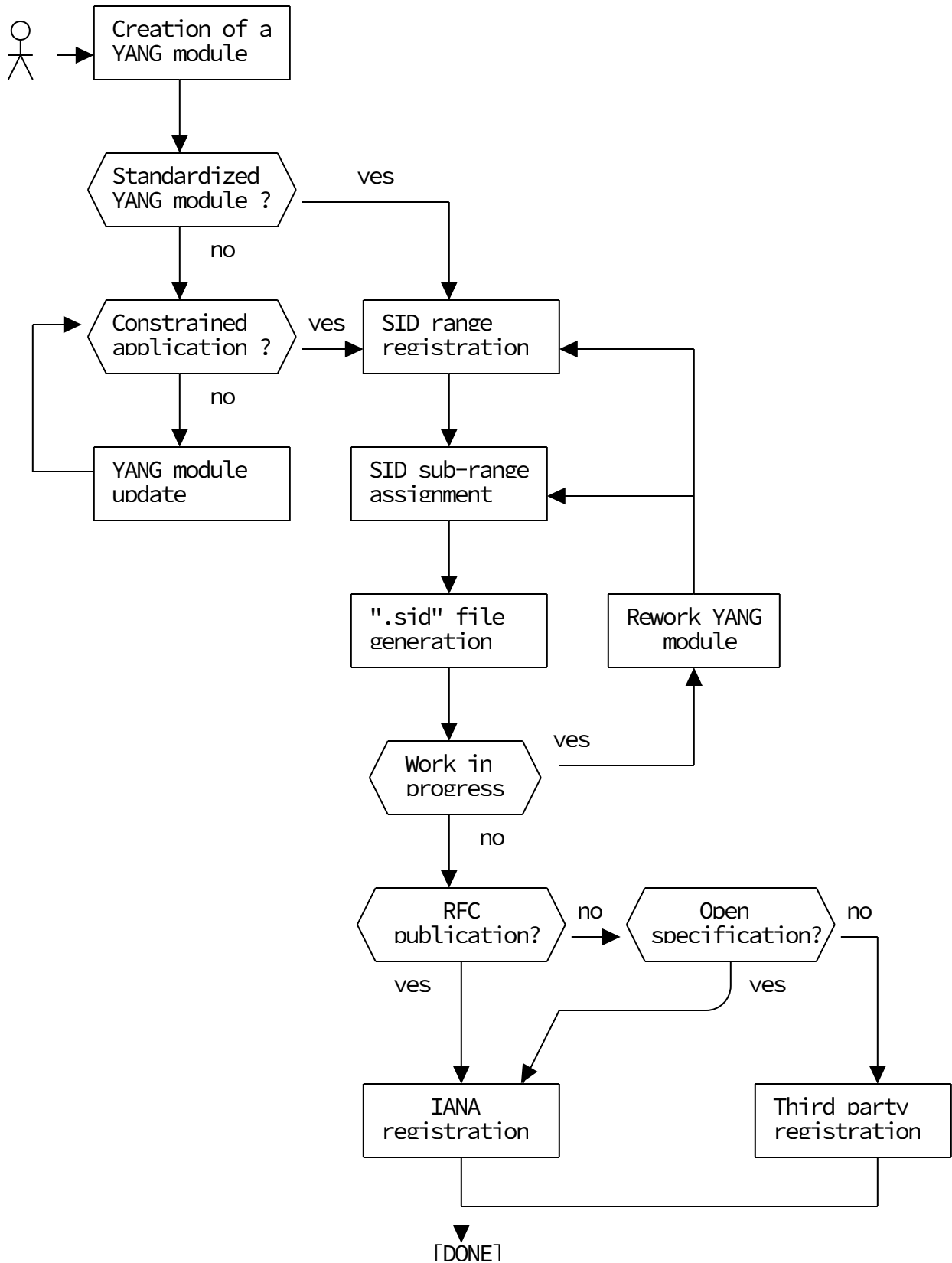The following activity diagram summarizes the creation of a YANG module and its associated ".sid" file.

Figure 4: SID Lifecycle

C.2. ".sid" File Update

The following Activity diagram summarizes the update of a YANG
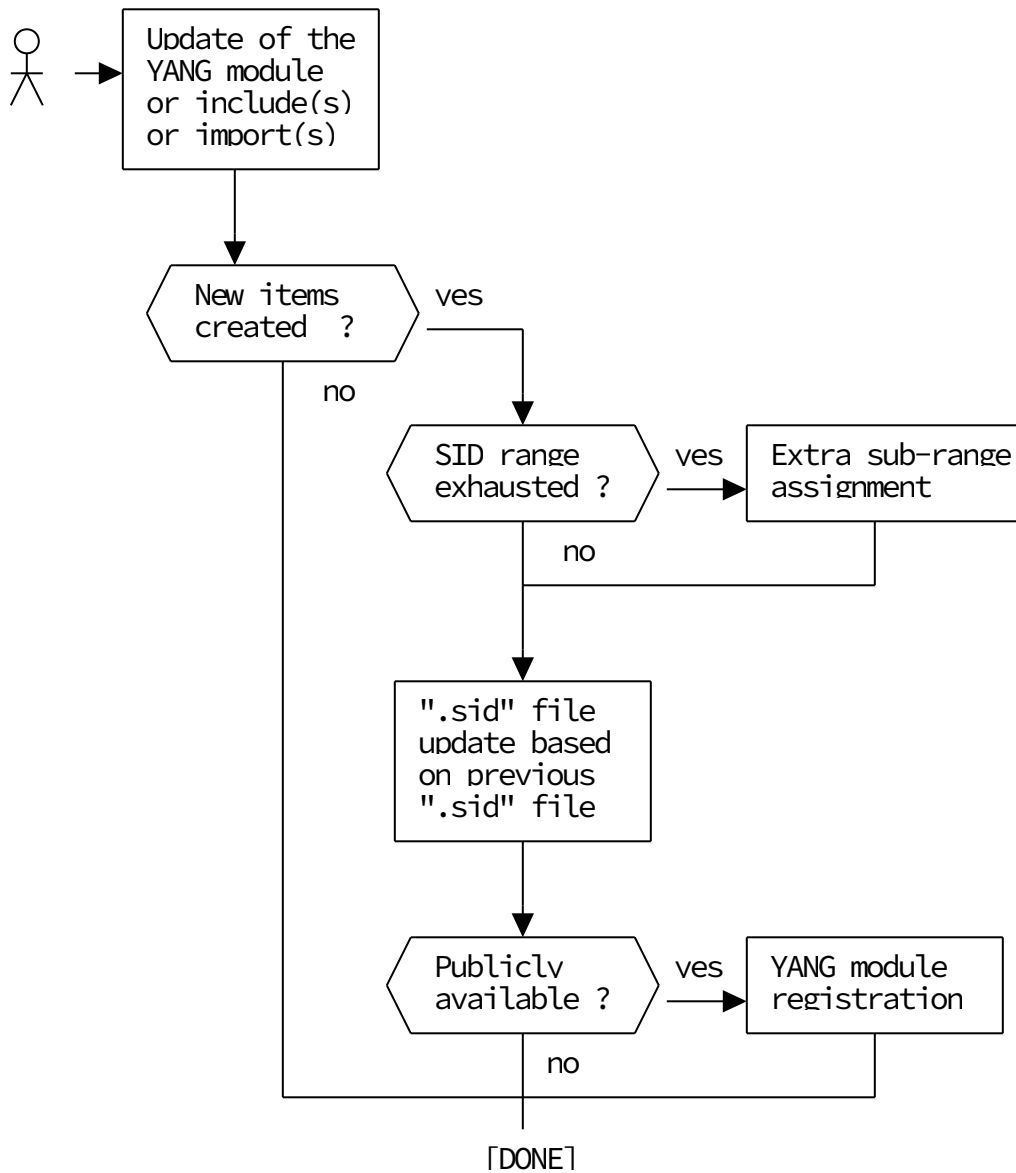module and its associated ".sid" file.

Figure 5: YANG and ".sid" file update

Contributors

Andy Bierman
YumaWorks
685 Cochran St.
Suite #160
Simi Valley, CA 93065
United States of America

Email: andy@yumaworks.com

Authors' Addresses

Michel Veillette (editor)
Trilliant Networks Inc.
610 Rue du Luxembourg
Granby Quebec J2J 2V2
Canada

Phone: +14503750556
Email: michel.veillette@trilliant.com

Alexander Pelov (editor)
Acklio
1137A avenue des Champs Blancs
35510 Cesson-Sevigne
France

Email: a@ackl.io

Ivaylo Petrov (editor)
Google Switzerland GmbH
Brandschenkestrasse 110
CH-8002 Zurich
Switzerland

Email: ivaylopetrov@google.com

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Michael Richardson
Sandelman Software Works

Email: mcr+ietf@sandelman.ca