Internet Engineering Task Force (IETF)
Request for Comments: 9595
Category: Standards Track
ISSN: 2070-1721

M. Veillette, Ed.
Trilliant Networks Inc.
A. Pelov, Ed.
IMT Atlantique
I. Petrov, Ed.
Google Switzerland GmbH
C. Bormann
Universitt Bremen TZI
M. Richardson
Sandelman Software Works

July 2024

YANG Schema Item iDentifier (YANG SID)

Abstract

YANG Schema Item iDentifiers (YANG SIDs) are globally unique 63-bit unsigned integers used to identify YANG items. SIDs provide a more compact method for identifying those YANG items that can be used efficiently, notably in constrained environments (RFC 7228). This document defines the semantics, registration processes, and assignment processes for 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

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9595.

Table of Contents

1. Introduction

Some of the items defined in YANG [RFC7950] require the use of a unique identifier. In both the 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 the 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 'rc:yang-data' extension [RFC8040] and the 'sx:structure' extension [RFC8791])

* 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 will use only a subset of the assigned SIDs; for example, for protocols other than NETCONF [RFC6241] that provide access to YANG-modeled data, such as [CORE-COMI], the transport 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 the Constrained YANG Module Library [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". Once they are considered "stable", SIDs are assigned permanently. Items introduced by a new revision of a YANG module are added to the list of SIDs already assigned. This is discussed in more detail in Section 2.

The assignment of SIDs to YANG items is 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 processes for YANG modules and associated SIDs. To enable the implementation of these processes, Section 4 defines a standard file format used to store and publish SIDs.

IETF-managed YANG modules that need to allocate SIDs will use the IANA mechanisms specified in this document. See Section 6 for details. YANG modules created by other parties allocate SID ranges using the IANA allocation mechanisms via Mega-Ranges (see Section 6.3); within the Mega-Range allocation, those other parties are free to make up their own mechanism.

Among other uses, YANG SIDs are particularly useful for obtaining a compact encoding for YANG-CBOR [RFC9254]. At the time of writing, a tool for automated ".sid" file generation is available as part of the open-source project PYANG [PYANG].

1.1. 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 [BCP14] (RFC 2119) (RFC 8174) 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

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.

YANG Schema Item iDentifier (YANG SID or simply SID): Unsigned integer used to identify different YANG items (cf. Section 3.2 of [RFC9254]).

YANG name: Text string used to identify different YANG items (cf. Section 3.3 of [RFC9254]).

2. Objectives

The overriding objective of the SID assignment and registration system is to ensure global interoperability of protocols that employ SIDs in order to communicate about data modeled in YANG. This objective poses certain requirements on the stability of SIDs while at the same time not hindering active evolution of the YANG modules the SIDs are intended to support.

Additional objectives include:

* enabling the developer of a YANG module to also be the originating entity for the SIDs pertaining to that module.

* making it easy for YANG developers to obtain SIDs.

* enabling other developers to define SIDs for a module where the developer of the module is not interested in assigning the SIDs.

* keeping an assignment regime that keeps short SIDs (2..4 bytes) readily available for the applications that would benefit from them while at the same time employing the vast 63-bit SID space to facilitate permissionless actions.

* enabling multiple entities to provide services that support the assignment of SIDs.

* maintaining some locality in the assignment of SIDs so the efficiencies of the SID delta mechanism can be fully employed.

* enabling various software components to operate in terms of SIDs without having complete information about other parties in the communication process.

While IANA ultimately maintains the registries that govern SIDs for IETF-defined modules, various support tools (such as, at the time of writing, the YANG Catalog [yangcatalog]) need to provide the support to enable SID assignment and use for modules still in IETF development. Developers of open-source or proprietary YANG modules also need to be able to serve as such entities autonomously, possibly forming alliances independent of the IETF, while still fitting in the overall SID number space managed by IANA. Obviously, this process has a number of parallels to the management of IP addresses but is also very different.

2.1. Technical Objectives

As discussed in the Introduction, SIDs are intended as globally unique (unsigned) integers.

Specifically, this means that:

Objective 1 (MUST): Any 63-bit unsigned integer either (1) is unassigned as a SID or (2) immutably maps to EXACTLY one YANG name. Only the transition from unassigned to that immutable mapping is defined.

This enables a recipient of a data structure employing SIDs to translate them into the globally meaningful YANG names that the existing encodings of YANG data such as YANG-XML [RFC7950] and YANG-JSON [RFC7951] employ today.

The term "YANG name" is not defined outside this document, and YANG has a complex system of names and entities that can have those names. Instead of defining the term technically, this set of objectives uses it in such a way that the overall objectives of YANG SID can be achieved.

A desirable objective is that:

Objective 2 (SHOULD): Any YANG name in active use has one SID assigned.

This means that:

1. There should not be YANG names without SIDs assigned.

2. YANG names should not have multiple SIDs assigned.

These objectives are unattainable in full, because YANG names are not necessarily born with a SID assignment and because entirely autonomous entities might decide to assign SIDs for the same YANG name without communicating ("like ships in the night"). Note that as long as this autonomy is maintained, any single observer will have the impression that Objective 2 is attained. Only when entities that have acted autonomously start communicating will a deviation be observed.

2.2. Module Evolution and Versioning

YANG modules evolve (see Section 11 of [RFC7950] and Section 4.27 of RFC 8407 [BCP216]). The technical objectives listed above are stated in terms that are independent of this evolution.

However, some modules are still in a very fluid state, and the assignment of permanent SIDs to the YANG names created in them is less desirable. This is true not only for new modules but also for emerging new revisions of existing stable modules.

Objective 3 (MUST): The SID management system is independent of any module versioning.

2.3. Solution Components and Derived Objectives

A registration system is used in order to guarantee the uniqueness of SIDs. To be able to provide some autonomy in allocation (and avoid information disclosure where it is not desirable), 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.

2.4. Parties and Roles

In the YANG development process, we can discern a number of parties

that are concerned with a YANG module:

module controller:
   The owner of the YANG module, i.e., the controller of the module's evolution.

registration entity:
   The controller of the module namespace, specifically also of the prefixes that are in common use. (This is not a required party.)

module repository:
   An entity that supplies modules to module users. This can be an "official" entity (e.g., IANA for IETF modules) or an "unofficial" entity (e.g., the YANG Catalog [yangcatalog]). Not all repositories are in a position to act as a registry, i.e., as a permanent record for the information they supply; these repositories need to recur to module owners as a stable source.

module user:
   An entity that uses a module, after obtaining it from the module controller or a module repository.

This set of parties needs to evolve to take on the additional roles that the SID assignment process requires:

SID assigner:
   An entity that assigns SIDs for a module. Objective 2 aims at having only one SID assigner for each module. SID assigners preferably stay the same over a module development process; however, this specification provides ".sid" files to ensure an organized handover.

SID range registry:
   An entity that supplies a SID assigner with SID ranges that it can use in assigning SIDs for a module. (In this specification, there is a structure with Mega-Ranges and individual SID ranges; this is not relevant here.)

SID repository:
   An entity that supplies SID assignments to SID users, usually in the form of a ".sid" file.

SID user:
   The module user that uses the SIDs provided by a SID assigner for a YANG module. SID users need to find SID assigners (or at least their SID assignments).

As the use of SIDs with YANG data models is introduced, the distribution of the SID roles to the existing parties for a YANG module will evolve.

The desirable end state of this evolution is shown in Table 1.

Role	Party
SID assigner	module developer
SID range registry	(as discussed in this specification)
SID repository	module repository
SID user	module user (naturally)

Table 1: Roles and Parties: Desired End State

This grouping of roles and parties puts the module developer in a position where it can achieve the objectives laid out in this section (a "type-1", "SID-guiding" module controller). (While a third party might theoretically assign additional SIDs and conflict with Objective 2, there is very little reason to do so if ".sid" files are always provided by the module developer with the module.)

The rest of this section is concerned with the transition to this end state.

For existing modules, there is no ".sid" file. The entity that stands in as the SID assigner is not specified. This situation has the highest potential for conflict with Objective 2.

Similarly, for new module development, the module owner may not have heard about SIDs or may not be interested in assigning them (e.g., because of lack of software or procedures within their organization).

For these two cases (which we will collectively call "type-3", "SID-oblivious" module controller), module repositories can act as a mediator, giving SID users access to a SID assigner that is carefully chosen to be a likely choice by other module repositories as well, maximizing the likelihood of achieving Objective 2.

If a module controller has heard about SIDs but is not assigning them yet, it can designate a SID assigner instead. This can lead to a stable, unique set of SID assignments being provided indirectly by a ("type-2", "SID-aware") module developer. Entities offering designated SID assigner services could make these available in an easy-to-use way, e.g., via a web interface.

The entity acting as a SID assigner minimally needs to record the SID range it uses for the SID assignment. If the SID range registry employed can record the module name and revision and if the assignment processes (including the software used) are stable, the SID assigner can theoretically reconstruct its assignments, but this could invite implementation bugs.

SID assigners attending to a module in development (not yet stable) need to decide whether SIDs for a new revision are reassigned from scratch ("clean slate") or use existing assignments from a previous revision as a base, only assigning new SIDs for new names. Once a module is declared stable, its SID assignments SHOULD be declared stable as well (except that, for existing YANG modules, some review may be needed before this is done).

This specification does not further discuss how mediating entities such as designated SID assigners or SID repositories could operate; instead, it supplies objectives for their operation.

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 the CoAP Management Interface (CORECONF) [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 in their YANG module. The assignments can then be stored in a ".sid" file. For an example of how this could be achieved, please refer to Appendix C.

Items introduced by a new revision of a YANG module are added to the list of SIDs already assigned. When this is done during the development of a new protocol document, it may be necessary to make provisional assignments. They may get changed, revised, or withdrawn during the development of a new standard. These provisional assignments are marked with a status of "unstable", so that they can be removed and the SID number possibly reassigned for a different YANG schema name/path later in the development process. When the specification is advanced to a final document, the assignment is marked with a status of "stable". During a period of development starting from a published specification, two variants of the ".sid" file should be made available by the tooling involved in that development: (1) a "published" ".sid" file with the existing stable SID assignments only (which the development effort should keep stable), as well as (2) an "unpublished" ".sid" file that also contains the unstable SID assignments.

Registration of the ".sid" file associated with a YANG module is optional but recommended; doing so will promote interoperability between devices and 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 possible scenario, please refer to the activity diagram shown in Figure 4 in Appendix C.

Each time a YANG module, one or more of its imported modules, or one or more of its included submodules are updated, a new ".sid" file MAY be created if the new or updated items will need SIDs. All the SIDs present in a previous version of the ".sid" file that was in active use 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’ item as defined in Section 4. These extra SIDs are used for subsequent assignments.

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

4. ".sid" File Format

".sid" files are used to persist and publish SIDs assigned to the different YANG items in a specific YANG module.

The following tree diagram [BCP215] provides an overview of the data model:

```
module: ietf-sid-file

  structure sid-file:
    +-- module-name            yang:yang-identifier
    +-- module-revision?       revision-identifier
    +-- sid-file-version?      sid-file-version-identifier
    +-- sid-file-status?       enumeration
    +-- 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]
     +-- status?             enumeration
     +-- namespace           enumeration
     +-- identifier          union
     +-- sid                 sid
```

Figure 1: YANG Tree for 'ietf-sid-file'

The following YANG module defines the structure of ".sid" files. Encoding is performed in JSON [STD90] using the rules defined in [RFC7951]. This module imports 'ietf-yang-types' [RFC6991] and 'ietf-yang-structure-ext' [RFC8791]. It also references [STD68], [RFC7950], and [BCP216].

```
<CODE BEGINS> file "ietf-sid-file@2024-07-31.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:alexander.pelov@imt-atlantique.fr>

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

  description
    "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 a YANG SID.

     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.

     Copyright (c) 2024 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 Revised 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 9595; see
   the RFC itself for full legal notices.";

revision 2024-07-31 {
  description
    "Initial revision.";
  reference
    "RFC 9595: YANG Schema Item iDentifier (YANG SID)";
}

typedef revision-identifier {
  type string {
    pattern '[0-9]{4}-[0-9]{2}-[0-9]{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 9595: 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:

     -  The leftmost (top-level) data node name is always in the
        namespace-qualified form.
     -  Any subsequent schema-node name is in the
        namespace-qualified form if the node is defined in a
        module other than its parent node.  Otherwise, the
        simple form is used.  No predicates are allowed.";
  reference
    "RFC 5234 (STD 68): Augmented BNF for Syntax Specifications:
                        ABNF
     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' file.";

  container sid-file {
    description
      "A wrapper container that together with the 'sx:structure'
       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
     represents the file structure of a '.sid' file.";

  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 - for instance, as the result
       of new processing or reported errata.";
  }

  leaf sid-file-status {
    type enumeration {
      enum unpublished {
        description
          "This '.sid' file is unpublished (BCP 216) and is
           also called a work-in-progress or workfile.
           This may be when it accompanies an unpublished YANG
           module or when only the '.sid' file itself is
```

```
          unpublished.
          The 'item' list MAY contain entries with a status
          value of 'unstable'.";
        reference
          "RFC 8407 (BCP 216): Guidelines for Authors and
                               Reviewers of Documents Containing
                               YANG Data Models";
      }
      enum published {
        description
          "This '.sid' file is published.  This status
           applies to '.sid' files for published YANG modules.
           The 'item' list MUST NOT contain entries with a
           status value of 'unstable'.";
      }
    }
    default "published";
    description
      "Optional leaf that specifies the status of the
       '.sid' file.";
  }

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

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

    description
      "Information about the revision used during the
       '.sid' file generation of each dependency, i.e., each
       YANG module that the YANG module associated with this
       '.sid' file imported.";

    leaf module-name {
      type yang:yang-identifier;
      description
        "YANG module name of this dependency.";
    }
    leaf module-revision {
      type revision-identifier;
      mandatory true;
      description
        "Revision of the YANG module of this dependency.";
    }
  }

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

       -  The first available value in the YANG-SID Range is
          'entry-point', and the last available value in the
          range is ('entry-point' + size - 1).
       -  The YANG-SID Ranges specified by all
          'assignment-range' entries 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 defines 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 status {
    type enumeration {
      enum stable {
        value 0;
        description
          "This SID allocation has been published as the
           stable allocation for the given namespace and
           identifier.";
      }
      enum unstable {
        value 1;
        description
          "This SID allocation has been done during a
           development process; it is not yet stable.";
      }
      enum obsolete {
        value 2;
        description
          "This SID allocation is no longer in use.  It is
           recorded to avoid reallocation of its SID value.";
      }
    }
    default "stable";
    description
      "The status field contains information about the
       stability of the allocation.  For each specific SID
       value, over time it can only transition from
       'unstable' to 'stable', and possibly from 'stable' to
       'obsolete'.";
  }

  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. 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 that are as authoritative as the YANG modules that they are using.

".sid" files are identified with and can employ _dereferenceable identifiers_, i.e., identifiers that could lead implementations in certain situations to automatically perform remote access, the target of which is indicated at least partially by those identifiers. This can give an attacker information from and/or control over such accesses, which can have security and privacy implications. Please also see Sections 6 and 7 of [DEREF-ID] for further considerations that may be applicable.

6. IANA Considerations

6.1. YANG Namespace Registration

This document registers the following XML namespace URN in the "IETF XML Registry", following the format defined in [BCP81]:

URI: 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 9595

6.2. ".sid" File Format Module Registration

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 9595

6.3. New IANA Registry: YANG-SID Mega-Ranges

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

6.3.1. 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, but in exceptional cases, it MAY be a multiple of 1,000,000.

* The policy of SID range allocations: Public, Private, or both.

* The contact information of the requesting organization, including the following:

  - Organization name.

  - URL.

6.3.2. Allocation Policy

The IANA policy for future additions to this registry is "Expert Review" (Section 4.5 of RFC 8126 [BCP26]).

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

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

  - The entry point of the allocated YANG-SID Range.

  - The size of the allocated YANG-SID Range.

  - Type: Public or Private.

    o Public ranges MUST include at least a reference to the YANG module and ".sid" files for that YANG-SID Range (e.g., see Section 6.4.3 as an example for how this is handled for the "IETF YANG-SID Ranges" registry).

    o Private ranges MUST be marked as "Private".

* A policy of allocation, which clearly identifies whether the YANG-SID Range allocations would be Private, Public, or both.

* Technical capacity to provide or refer to ".sid" files in a way that meets the security objective of data integrity for these files (see also Section 5).

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

6.3.2.1. First Allocation

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

6.3.2.2. Consecutive Allocations

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

If such a request for an additional allocation is made within 3 years of 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.

6.3.3. Initial Contents of the Registry

This registry contains the following initial entry:

Entry Point	Size	Allocation Name	Organization	URL

0	1,000,000	Public	IANA	<https://www.iana.org/>

Table 2: YANG-SID Mega-Ranges Registry: Initial Assignment

6.4. New IANA Registry: IETF YANG-SID Ranges

The name of this registry is "IETF YANG-SID Ranges". This registry is used to record information related to the assignment of SID Ranges (e.g., entry point and size) to YANG modules identified by module name.

6.4.1. Structure

Each entry in this registry must include:

* The SID range entry point.
* The SID range size.
* The YANG module name.
* A document reference (the document making the registration).

6.4.2. Allocation Policy

The first million SIDs assigned to IANA are subdivided as follows:

1. The range of 0 to 999 (size 1,000) is subject to "IESG Approval" as defined in Section 4.10 of RFC 8126 [BCP26]; 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.
2. The ranges of 1,000 to 59,999 (size 59,000) and 100,000 to 299,999 (size 200,000) are designated for YANG modules defined in RFCs.
    a. The IANA policy for additions to this registry is:
        i. "Expert Review" (Section 4.5 of RFC 8126 [BCP26]) if the ".sid" file comes from a YANG module from an existing RFC.
        ii. "RFC Required" (Section 4.7 of RFC 8126 [BCP26]) otherwise.
    b. 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 an RFC (Early Allocation with a request from the working group chairs as defined by [BCP100]).
3. 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 and their interoperability is therefore limited. The IANA policy for this range is "Experimental Use" (Section 4.2 of RFC 8126 [BCP26]).
4. The range of 300,000 to 999,999 (size 700,000) is "Reserved" as defined in Section 6 of RFC 8126 [BCP26].

Entry Point	Size	IANA Policy

0	1,000	IESG Approval
1,000	59,000	RFC and Expert Review Required (see item 2 above)
60,000	40,000	Reserved for Experimental Use
100,000	200,000	RFC and Expert Review Required (see item 2 above)
300,000	700,000	Reserved

Table 3: IETF YANG-SID Ranges Registry: Policies for IETF Ranges

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 assignments within the same range of new YANG items introduced by subsequent revisions. The SID range size SHOULD NOT exceed 1,000; 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 a slightly less efficient representation.

If a SID range is allocated for an existing RFC through the "Expert Review" policy (Section 4.5 of RFC 8126 [BCP26]), the Reference field for the given allocation should point to the RFC that the YANG module is defined in.

If 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 for the "RFC Required" range (Section 2 of RFC 7120 [BCP100]).

6.4.3. Publication of the ".sid" File

During an RFC's publication process, IANA contacts the designated expert team ("the team"), who are responsible for delivering a final ".sid" file for each module defined by the RFC. For a type-3 developer (SID-oblivious; see Section 2.4), the team creates a new ".sid" file from each YANG module; see below. For a type-2 (SID-aware) developer, the team first obtains the existing draft ".sid" file from a stable reference in the approved draft; for a type-1 (SID-guiding) developer, the team extracts the ".sid" file from the approved draft.

The team uses a tool to generate a final ".sid" file from each YANG module; the final ".sid" file has all SID assignments set to "stable" and the ".sid" file status set to "published". A published ".sid" file MUST NOT contain SID assignments with a status of "unstable".

For the cases other than type-3 (SID-oblivious), the team feeds the existing draft ".sid" file as an input ("reference ".sid" file") to the tool so that the changes resulting from regeneration are minimal. For YANG modules that are revisions of previously published modules, any existing published ".sid" file needs to serve as a reference ".sid" file for the tool, during generation of either the revised draft ".sid" file (type-1, type-2) or the final ".sid" file (type-3).

In any case, the team checks the generated file, including checking for validity as a ".sid" file, for consistency with the SID range allocations, for full coverage of the YANG items in the YANG module,

and for the best achievable consistency with the existing draft ".sid" file.

The designated experts then give the ".sid" file to IANA to publish in the "IETF YANG-SID Modules" registry (Section 6.5) 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 to it is to be inserted in the RFC. (Note that the present RFC is an exception to this rule, as the ".sid" file also serves as an example for exposition.) Internet-Drafts that need SIDs assigned to their new modules for use in the text of the document, e.g., for examples, need to alert the RFC Editor in the draft text that this is the case. Such RFCs cannot be produced by type-3 (SID-oblivious) developers: the SIDs used in the text need to be assigned in the existing draft ".sid" file, and the designated expert team needs to check that the assignments in the final ".sid" file are consistent with the usage in the RFC text or that the approved draft text is changed appropriately.

6.4.4. Initial Contents of the Registry

Initial entries in this registry are as follows:

Entry Point	Size	Module Name	Reference
0	1	(Reserved: not a valid SID)	RFC 9595
1,000	100	ietf-coreconf	RFC 9595, [CORE-COMI]
1,100	50	ietf-yang-types	[RFC6991]
1,150	50	ietf-inet-types	[RFC6991]
1,200	50	iana-crypt-hash	[RFC7317]
1,250	50	ietf-netconf-acm	[STD91]
1,300	50	ietf-sid-file	RFC 9595
1,500	100	ietf-interfaces	[RFC8343]
1,600	100	ietf-ip	[RFC8344]
1,700	100	ietf-system	[RFC7317]
1,800	400	iana-if-type	[RFC7224]

Table 4: IETF YANG-SID Ranges Registry: Initial Range Assignments

For allocation, RFC publication of the YANG module is required as per the "RFC Required" policy defined in Section 4.7 of RFC 8126 [BCP26]. The YANG module must be registered in the "YANG Module Names" registry according to the rules specified in Section 14 of [RFC6020].

6.5. New IANA Registry: IETF YANG-SID Modules

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

6.5.1. 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 URI for the associated ".yang" file. This file link must be present in the "File" column of the "YANG Module Names" registry.

* The URI for the ".sid" file that defines the allocation. The ".sid" file is stored by IANA.

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

6.5.2. Allocation Policy

The allocation policy is "Expert Review" (Section 4.5 of RFC 8126 [BCP26]). 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 this document.

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

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

6.5.3. Recursive Allocation of YANG SIDs 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 range allocations using Early Allocations [BCP100]. The details of the Early Allocation to be requested, including the timeline envisioned, 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 6.4.2) for their SID allocations or other values that do not create ambiguity with other SID uses (for example, they can use ranges and SIDs registered in a non-IANA-managed registry that is based on a YANG-SID Mega-Range assignment).

After working group adoption, any modification of a ".sid" file is expected to be discussed on the mailing lists of the appropriate working groups. Specific attention should be paid to implementers' opinions 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.

As the concept of SIDs is first used, 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, in a process that will generally be analogous to that in Section 6.4.3 for the type-3 (SID-oblivious) case.

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.

* If the document is an unprocessed Internet-Draft adopted in a working group, then an Early Allocation is performed for this document as well. Early Allocations require approval by an IESG area director. An Early Allocation that requires additional allocations will list the other allocations in its description and will be cross-posted to the mailing lists of any other working groups concerned.

* A YANG module that references a module in a document that 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 Section 3 of RFC 7120 [BCP100], a shepherding AD will stand in for the working group chairs if the document is not the product of an IETF working group, effectively allowing the AD to exempt a document from this policy.

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

[BCP100] defines a time limit for the validity of Early Allocations, after which they expire unless they are renewed. Section 3.3 of RFC 7120 [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 must not be considered to have expired while the document is under IESG consideration or is awaiting publication in the RFC Editor's queue after approval by the IESG.

6.5.4. Initial Contents of the Registry

At the time of writing, this registry does not contain any entries.

6.6. Media Type and Content-Format Registration

6.6.1. Media Type application/yang-sid+json

This document adds the following media type to the "Media Types" registry.

Name	Template	Reference
yang-sid+json	application/yang-sid+json	RFC 9595

Table 5: ".sid" File Media Type Registration

Type name: application

Subtype name: yang-sid+json

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary (UTF-8)

Security considerations: See Section 5 of RFC 9595.

Published specification: RFC 9595

Applications that use this media type: Applications that need to obtain YANG SIDs to interchange YANG-modeled data in a concise and efficient representation.

Fragment identifier considerations: The syntax and semantics of fragment identifiers specified for "application/yang-sid+json" is as specified for "application/json". (At publication of this document, there is no fragment identification syntax defined for "application/json".)

Additional information:

Magic number(s): N/A
File extension(s): .sid
Macintosh file type code(s): N/A

Person & email address to contact for further information: CORE WG mailing list (core@ietf.org) or IETF Applications and Real-Time Area (art@ietf.org)

Intended usage: COMMON

Restrictions on usage: none

Author/Change controller: IETF

6.6.2. CoAP Content-Format

This document adds the following Content-Format to the "CoAP Content-Formats" registry within the "Constrained RESTful Environments (CoRE) Parameters" group of registries, where 260 has been assigned from the "IETF Review" (256-9,999) range (see Section 4.8 of RFC 8126 [BCP26]).

Content Type	Content Coding	ID	Reference
application/yang-sid+json	-	260	RFC 9595

Table 6: ".sid" File Content-Format Registration

7. References

7.1. Normative References

[BCP100] Best Current Practice 100, <https://www.rfc-editor.org/info/bcp100>. At the time of writing, this BCP comprises the following:

Cotton, M., "Early IANA Allocation of Standards Track Code Points", BCP 100, RFC 7120, DOI 10.17487/RFC7120, January

2014, <https://www.rfc-editor.org/info/rfc7120>.

[BCP14] Best Current Practice 14,
<https://www.rfc-editor.org/info/bcp14>.
At the time of writing, this BCP comprises the following:

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>.

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>.

[BCP81] Best Current Practice 81,
<https://www.rfc-editor.org/info/bcp81>.
At the time of writing, this BCP comprises the following:

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>.

[RFC8791] Bierman, A., Bjrklund, M., and K. Watsen, "YANG Data
Structure Extensions", RFC 8791, DOI 10.17487/RFC8791,
June 2020, <https://www.rfc-editor.org/info/rfc8791>.

[STD68] Internet Standard 68,
<https://www.rfc-editor.org/info/std68>.
At the time of writing, this STD comprises the following:

Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax
Specifications: ABNF", STD 68, RFC 5234,
DOI 10.17487/RFC5234, January 2008,
<https://www.rfc-editor.org/info/rfc5234>.

[STD90] Internet Standard 90,
<https://www.rfc-editor.org/info/std90>.
At the time of writing, this STD comprises the following:

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>.

7.2. Informative References

[BCP215] Best Current Practice 215,
<https://www.rfc-editor.org/info/bcp215>.
At the time of writing, this BCP comprises the following:

Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[BCP216] Best Current Practice 216, <https://www.rfc-editor.org/info/bcp216>. At the time of writing, this BCP comprises the following:

Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[BCP26] Best Current Practice 26, <https://www.rfc-editor.org/info/bcp26>. At the time of writing, this BCP comprises the following:

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>.

[CORE-COMI] Veillette, M., Ed., van der Stok, P., Ed., Pelov, A., Ed., Bierman, A., and C. Bormann, Ed., "CoAP Management Interface (CORECONF)", Work in Progress, Internet-Draft, draft-ietf-core-comi-18, 23 July 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-core-comi-18>.

[DEREF-ID] Bormann, C. and C. Amsss, "The ’dereferenceable identifier’ pattern", Work in Progress, Internet-Draft, draft-bormann-t2trg-deref-id-03, 2 March 2024, <https://datatracker.ietf.org/doc/html/draft-bormann-t2trg-deref-id-03>.

[PYANG] Bjrklund, M., "An extensible YANG validator and converter in python", commit fc9a965, May 2024, <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>.

[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>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[RFC9195] Lengyel, B. and B. Claise, "A File Format for YANG Instance Data", RFC 9195, DOI 10.17487/RFC9195, February 2022, <https://www.rfc-editor.org/info/rfc9195>.

[RFC9254] Veillette, M., Ed., Petrov, I., Ed., Pelov, A., Bormann, C., and M. Richardson, "Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR)", RFC 9254, DOI 10.17487/RFC9254, July 2022, <https://www.rfc-editor.org/info/rfc9254>.

[STD91] Internet Standard 91, <https://www.rfc-editor.org/info/std91>. At the time of writing, this STD comprises the following:

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>.

[YANG-LIBRARY] Veillette, M., Ed. and I. Petrov, Ed., "Constrained YANG Module Library", Work in Progress, Internet-Draft, draft-ietf-core-yang-library-03, 11 January 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-core-yang-library-03>.

[yangcatalog] "YANG Catalog", <https://yangcatalog.org>.

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 [STD91])

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

For purposes of exposition, per [RFC8792], line breaks have been introduced below in some JSON strings that represent overly long identifiers.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-sid-file:sid-file": {
    "module-name": "ietf-system",
    "module-revision": "2014-08-06",
    "description": "Example '.sid' file",
```

```
"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"
  }
],
"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",
  "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/input",
  "sid": "1775"
},
{
  "namespace": "data",
  "identifier": "/ietf-system:set-current-datetime/input/\
                                          current-datetime",
  "sid": "1776"
},
{
  "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"
},
{
  "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",
  "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",
  "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)

Appendix B. SID Autogeneration

The 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' entries are sorted in descending order, and 'identifier' entries are sorted 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 the 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' list item should reflect the revision numbers of each YANG module that the YANG module imports at the moment of file generation.

When updating a YANG module that is in active use, the existing SID assignments are maintained. (In contrast, when evolving an early

version of an Internet-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 for allowing 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 the 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 to be 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 to ".sid" files can also be automated using the same method as that described above, except that in step #3, previous SID assignments are kept, only previously unassigned YANG items are processed, and these are assigned previously unassigned SIDs. 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 either (1) should be 0 or (2) should not be present.

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

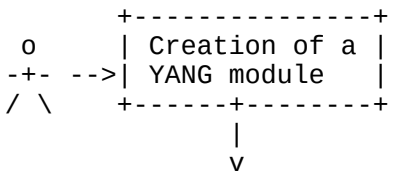
Appendix C. ".sid" File Lifecycle

Before assigning SIDs to their YANG modules, YANG module authors must acquire a SID range from a registry of YANG-SID Ranges. If the YANG module is part of an IETF Internet-Draft or RFC, the SID range needs to be acquired from the "IETF YANG-SID Ranges" registry as defined in Section 6.4. For the other YANG modules, the authors can choose to acquire a SID range from any registry of YANG-SID Ranges.

Once the SID range is acquired, owners can use it to generate one or more ".sid" files for their YANG module or modules. It is recommended to leave some unallocated SIDs following the allocated range in each ".sid" file in order to allow better evolution of the owners' YANG modules 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.



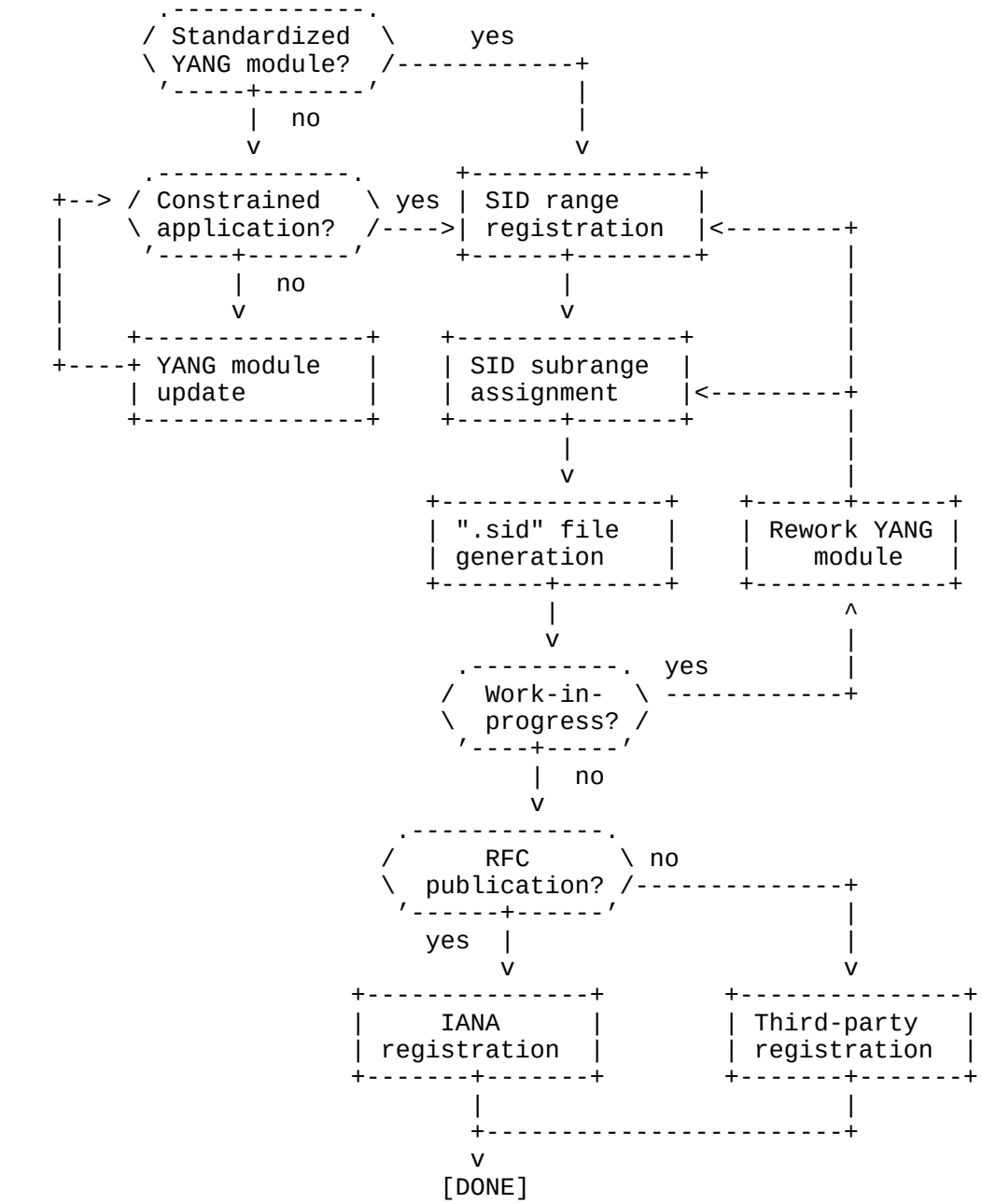
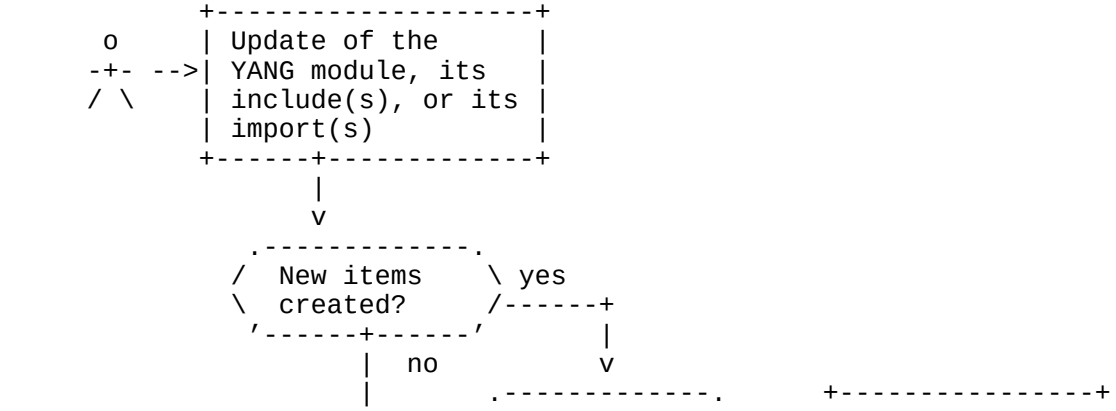


Figure 4: SID Lifecycle

C.2. ".sid" File Update

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



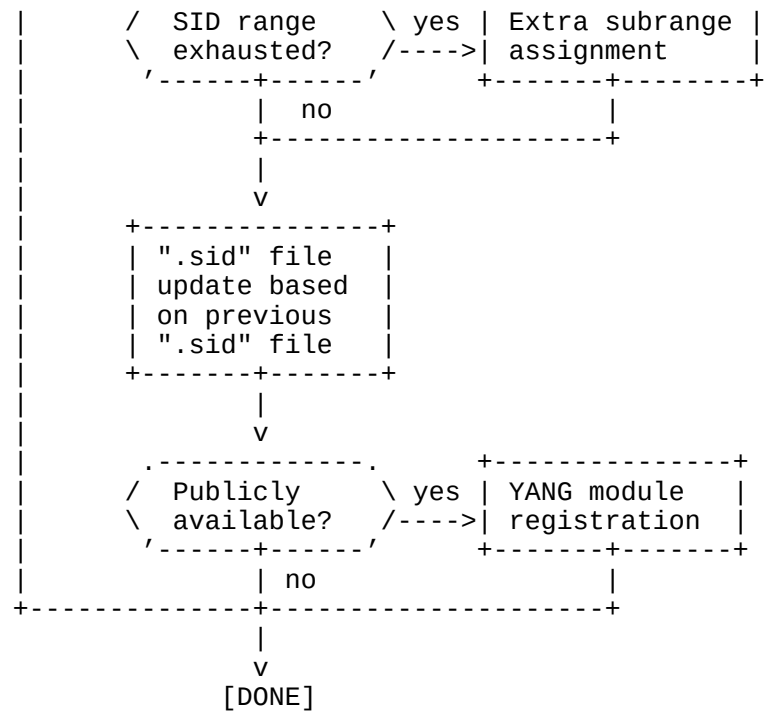


Figure 5: YANG and ".sid" File Update

Appendix D. Keeping a ".sid" File in a YANG Instance Data File

[RFC9195] defines a format for "YANG instance data". This essentially leads to an encapsulation of the instance data within some metadata envelope.

If a ".sid" file needs to be stored in a YANG instance data file, this can be achieved by embedding the value of the ".sid" file as the value of the content-data member in the following template and copying over the second-level members as indicated with the angle brackets:

```
{
  "ietf-yang-instance-data:instance-data-set": {
    "name": "<module-name>@<module-revision>.sid",
    "description":  ["<description>"],
    "content-schema": {
      "module": "ietf-sid-file@2024-06-17"
    },
    "content-data": {  <replace this object>
      "ietf-sid-file:sid-file" : {
        "module-name": ...
      }
    }
  }
}
```

Acknowledgments

The authors would like to thank Andy Bierman, Abhinav Somaraju, Peter van der Stok, Laurent Toutain, and Randy Turner for their help during the development of this document and their useful comments during the review process. Special thanks go to the IESG members who supplied very useful comments during the IESG processing phase, in particular to Benjamin Kaduk and Rob Wilton, and to Francesca Palombini as responsible AD.

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: +1-450-375-0556
Email: michel.veillette@trilliant.com

Alexander Pelov (editor)
IMT Atlantique
2 rue de la Chtaigneraie
35510 Cesson-Svign Cedex
France
Email: alexander.pelov@imt-atlantique.fr

Ivaylo Petrov (editor)
Google Switzerland GmbH
Brandschenkestrasse 110
CH-8002 Zurich
Switzerland
Email: ivaylopetrov@google.com

Carsten Bormann
Universitt Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org

Michael Richardson
Sandelman Software Works
Canada
Email: mcr+ietf@sandelman.ca