Internet Engineering Task Force (IETF)
Request for Comments: 6643
Category: Standards Track
ISSN: 2070-1721

J. Schoenwaelder
Jacobs University

July 2012

Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules

Abstract

YANG is a data modeling language used to model configuration and state data manipulated by the Network Configuration Protocol (NETCONF), NETCONF remote procedure calls, and NETCONF notifications. The Structure of Management Information (SMIv2) defines fundamental data types, an object model, and the rules for writing and revising MIB modules for use with the Simple Network Management Protocol (SNMP). This document defines a translation of SMIv2 MIB modules into YANG modules, enabling read-only (config false) access to data objects defined in SMIv2 MIB modules via NETCONF.

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 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6643.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction

This document describes a translation of SMIv2 [RFC2578], [RFC2579], [RFC2580] MIB modules into YANG [RFC6020] modules, enabling read-only (config false, as defined in Section 7.19.1 of RFC 6020) access to SMIv2 objects defined in SMIv2 MIB modules via NETCONF [RFC6241]. For a discussion why SMIv2 read-write or read-create objects are translated to read-only (config false) YANG objects, see Section 11.

YANG modules generated from SMIv2 modules should not be modified. Any necessary changes should be made by modifying the original SMIv2 modules (with proper updates of the SMIv2 LAST-UPDATED and REVISION clauses) and then running the translation defined in this memo again. Note that this does not affect the usage of YANG augments and or YANG deviations: YANG modules generated from SMIv2 modules can be augmented like any other YANG module, and YANG deviations can be used to document how an implementation deviates from the generated YANG module.

SMIv1 modules can be converted to YANG by first following the rules in [RFC3584] to convert the SMIv1 module to SMIv2 and then following the rules in this document to convert the obtained SMIv2 module to YANG.

The SMIv2-to-YANG mapping is illustrated by examples showing the translation of parts of the IF-MIB [RFC2863], the DIFFSERV-MIB [RFC3289], and the RMON2-MIB [RFC4502] SMIv2 modules.

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].

2. Mapping of Well-Known Types

The SMIv2 base types and some well-known derived textual conventions are mapped to YANG types according to Appendix A. The mapping of the OCTET STRING depends on the context. If an OCTET STRING type has an associated DISPLAY-HINT, then the corresponding YANG base type is the string type. An implementation MUST format an OCTET STRING value according to the DISPLAY-HINT, as described in RFC 2579. If an OCTECT STRING type does not have an associated DISPLAY-HINT, the binary type is used. Similarly, the mapping of the INTEGER type depends on its usage as an enumeration or a 32-bit integral type. Implementations should provide implementation-specific options to handle situations where DISPLAY- HINTs are added during a revision of a module and backwards compatibility must be preserved, i.e., an added DISPLAY-HINT needs to be ignored.

The mappings shown in Appendix A may require to import the ietf-yang-types, ietf-inet-types, or ietf-yang-smiv2 YANG modules since some SMIv2 types and textual conventions map to YANG types defined in the ietf-yang-types and ietf-inet-types YANG modules defined in [RFC6021] and the ietf-yang-smiv2 YANG module defined in this document. Implementations MUST add any additional imports required by the type mapping.

3. Translation of SMIv2 Modules and SMIv2 IMPORT Clauses

SMIv2 modules are mapped to corresponding YANG modules. The generated YANG module name MUST be the same as the SMIv2 module name.

The YANG namespace MUST be constructed out of the IANA-registered prefix urn:ietf:params:xml:ns:yang:smiv2: (see Section 12) followed by the SMIv2 module name. Since SMIv2 module names can be assumed to be unique (see Section 3 in [RFC2578]), the resulting YANG namespace is unique.

The YANG prefix MAY be derived from the SMIv2 module name using the module prefix generation algorithm described in Appendix B. The YANG prefix is supposed to be short, and it must be unique within the set of all prefixes used by a YANG module. The algorithm described in Appendix B generates such prefixes.

SMIv2 IMPORT clauses are translated to YANG import statements. One major difference between the SMIv2 import mechanism and the YANG import mechanism is that SMIv2 IMPORT clauses import specific symbols from an SMIv2 module, while the YANG import statement imports all symbols of the referenced YANG module.

In order to produce correct and complete YANG import statements, the following rules MUST be used:

- Process each item in each SMIv2 IMPORT clause as follows:

  1. If an import statement for this SMIv2 module has already been generated, then ignore this item.

  2. Otherwise, if the SMIv2 module name is SNMPv2-SMI or SNMPv2-CONF, then ignore this item. Note that these two modules can be completely ignored since all definitions in these modules are translated by translation rules.

  3. Otherwise, if this item is a textual convention matching one of the textual conventions in the SMIv2 types column of Appendix A (e.g., MacAddress, PhysAddress, or TimeStamp) then ignore this item.

4. Otherwise, if the item is used in a SYNTAX clause of an OBJECT-TYPE whose MAX-ACCESS is not accessible-for-notify, then generate an import statement as described below.

5. Otherwise, if the item is used in an OBJECTS clause of a NOTIFICATION-TYPE, then generate an import statement as described below.

6. Otherwise, if the item is used in an INDEX or AUGMENTS clause, then generate an import statement as described below.

7. Otherwise, ignore this item. Some examples of this case are OBJECT IDENTIFIER assignments and objects that are only referenced in MODULE-COMPLIANCE, OBJECT-GROUP, or NOTIFICATION-GROUP clauses.

- Generate any additional import statements as required by the type translations according to the type mapping table Appendix A. This requires the translator to consider all the types used in the SMIv2 module in order to produce the imports.

- Generate an import statement for the YANG module ietf-yang-smiv2 with the prefix smiv2.

The generated import statements use the untranslated SMIv2 module names or the names of well-known YANG modules as their argument. The import statement must contain a prefix statement. The prefixes MAY be generated by applying the module prefix generation algorithm described in Appendix B.

3.1. Example: IMPORTS of IF-MIB

The translation of the IF-MIB [RFC2863] leads to the YANG module and namespace/prefix statement and the import statements shown below. The prefix is the translation of the SMIv2 module name IF-MIB to lowercase (consisting of two tokens and thus no further abbreviation).

```
  module IF-MIB {

    namespace "urn:ietf:params:xml:ns:yang:smiv2:IF-MIB";
    prefix "if-mib";

    import IANAifType-MIB      { prefix "ianaiftype-mib"; }
    import SNMPv2-TC           { prefix "snmpv2-tc"; }
    import ietf-yang-types     { prefix "yang"; }
    import ietf-yang-smiv2     { prefix "smiv2"; }
  }
```

4. Translation of the MODULE-IDENTITY Macro

SMIv2 requires an invocation of the MODULE-IDENTITY macro to provide contact and revision history for a MIB module. The clauses of the SMIv2 MODULE-IDENTITY macro MUST be translated into YANG statements as detailed below.

4.1. MODULE-IDENTITY Translation Rules

- The SMIv2 ORGANIZATION clause is mapped to the YANG organization statement.
- The SMIv2 CONTACT-INFO clause is mapped to the YANG contact statement.
- The SMIv2 DESCRIPTION clause is mapped to the YANG description statement.
- Each SMIv2 REVISION clause is mapped to a YANG revision statement. The revision is identified by the date argument of the SMIv2 REVISION clause. DESCRIPTION sub-clauses of REVISION clauses are mapped to corresponding description statement nested in revision clauses.
- The SMIv2 LAST-UPDATED clause is ignored if the associated date matches a REVISION clause. Otherwise, an additional revision statement is generated.
- A top-level YANG container is generated. The container’s name is the SMIv2 module name, and the container MUST be config false. The generation of the top-level container MAY be skipped if the SMIv2 module does not define any objects that go into the top-level container (e.g., an SMIv2 module only defining textual conventions).
- The object identifier value of the invocation of the SMIv2 MODULE-IDENTITY is translated into an smiv2:oid statement contained in an smiv2:alias statement representing the MODULE-IDENTITY macro invocation. Refer to the YANG extension defined in Section 10.

While all proper SMIv2 modules must have exactly one MODULE-IDENTITY macro invocation, there are a few notable exceptions. The modules defining the SMIv2 language (i.e., the SNMPv2-SMI, SNMPv2-TC, and SNMPv2-CONF modules) do not invoke the MODULE-IDENTITY macro. Furthermore, SMIv2 modules generated from SMIv1 modules may miss an invocation of the MODULE-IDENTITY macro as well. In such cases, it is preferable to not generate organization, contact, description, or revision statements.

4.2. Example: MODULE-IDENTITY of IF-MIB

The translation of the MODULE-IDENTITY of the IF-MIB [RFC2863] leads to the following YANG statements:

```
  organization
   "IETF Interfaces MIB Working Group";

  contact
   "Keith McCloghrie
    Cisco Systems, Inc.
    170 West Tasman Drive
    San Jose, CA  95134-1706
    US

    408-526-5260
    kzm@cisco.com";

  description
   "The MIB module to describe generic objects for network
    interface sub-layers.  This MIB is an updated version of
    MIB-II's ifTable, and incorporates the extensions defined in
    RFC 1229.";

  revision "2000-06-14" {
    description
     "Clarifications agreed upon by the Interfaces MIB WG, and
      published as RFC 2863.";
  }
  revision "1996-02-28" {
    description
     "Revisions made by the Interfaces MIB WG, and published in
      RFC 2233.";
  }
  revision "1993-11-08" {
    description
     "Initial revision, published as part of RFC 1573.";
  }

  container IF-MIB {
    config false;
  }
```

5. Translation of the TEXTUAL-CONVENTION Macro

The SMIv2 uses invocations of the TEXTUAL-CONVENTION macro to define new types derived from the SMIv2 base types. Invocations of the TEXTUAL-CONVENTION macro MUST be translated into YANG typedef statements as detailed below.

5.1. TEXTUAL-CONVENTION Translation Rules

The name of the TEXTUAL-CONVENTION macro invocation is used as the name of the generated typedef statement. The clauses of the SMIv2 TEXTUAL-CONVENTION macro are mapped to YANG statements embedded in the typedef statement as follows:

- The SMIv2 DISPLAY-HINT clause is used to determine the type mapping of types derived form the OCTET STRING type as explained in Section 2. Furthermore, the DISPLAY-HINT value MAY be used to generate a regular expression for the YANG pattern statement within the type statement.

- The SMIv2 DISPLAY-HINT is translated into an smiv2:display-hint statement. Refer to the YANG extension defined in Section 10.

- The SMIv2 STATUS clause is mapped to the YANG status statement. The generation of the YANG status statement is skipped if the value of the STATUS clause is current.

- The SMIv2 DESCRIPTION clause is mapped to the YANG description statement.

- The SMIv2 REFERENCE clause is mapped to the YANG reference statement.

- The SMIv2 SYNTAX clause is mapped to the YANG type statement. SMIv2 range restrictions are mapped to YANG range statements, while SMIv2 length restrictions are mapped to YANG length statements. SMIv2 INTEGER enumerations are mapped to YANG enum/value statements. SMIv2 BITS are mapped to YANG bit/position statements. For OCTET STRING types that are mapped to a YANG string base type (see Section 2), the length specified in the YANG length statement must be consistent with the stringified representation of values. If an implementation is unable to derive a proper length restrictions, then the YANG length statement MUST be omitted.

This translation assumes that labels of named numbers and named bits do not change when an SMIv2 module is revised. This is consistent with the clarification of the SMIv2 module revision rules in Section 4.9 of [RFC4181].

5.2. Example: OwnerString and InterfaceIndex of IF-MIB

The translations of the OwnerString and InterfaceIndex textual conventions of the IF-MIB [RFC2863] are shown below.

```
  typedef OwnerString {
    type string {
      length "0..255";
      pattern '\p{IsBasicLatin}{0,255}';
    }
    status deprecated;
    description
     "This data type is used to model an administratively
      assigned name of the owner of a resource.  This information
      is taken from the NVT ASCII character set.  It is suggested
      that this name contain one or more of the following: ASCII
      form of the manager station's transport address, management
      station name (e.g., domain name), network management
      personnel's name, location, or phone number.  In some cases
      the agent itself will be the owner of an entry.  In these
      cases, this string shall be set to a string starting with
      'agent'.";
    smiv2:display-hint "255a";
  }

  typedef InterfaceIndex {
    type int32 {
      range "1..2147483647";
    }
    description
     "A unique value, greater than zero, for each interface or
      interface sub-layer in the managed system.  It is
      recommended that values are assigned contiguously starting
      from 1.  The value for each interface sub-layer must remain
      constant at least from one re-initialization of the entity's
      network management system to the next re-initialization.";
    smiv2:display-hint "d";
  }
```

5.3. Example: IfDirection of the DIFFSERV-MIB

The translation of the IfDirection textual convention of the DIFFSERV-MIB [RFC3289] is shown below.

```
  typedef IfDirection {
    type enumeration {
      enum inbound  { value 1; }
      enum outbound { value 2; }
    }
    description
     "IfDirection specifies a direction of data travel on an
      interface. 'inbound' traffic is operated on during reception
      from the interface, while 'outbound' traffic is operated on
      prior to transmission on the interface.";
  }
```

6. Translation of OBJECT IDENTIFIER Assignments

The SMIv2 uses OBJECT IDENTIFIER assignments to introduce names for intermediate nodes in the OBJECT IDENTIFIER tree. OBJECT IDENTIFIER assignments are translated into smiv2:alias statements. Refer to the YANG extension defined in Section 10.

7. Translation of the OBJECT-TYPE Macro

The SMIv2 uses the OBJECT-TYPE macro to define objects and the structure of conceptual tables. Objects exist either as scalars (exactly one instance within an SNMP context) or columnar objects within conceptual tables (zero or multiple instances within an SNMP context). A number of auxiliary objects define the index (key) of a conceptual table. Furthermore, conceptual tables can be augmented by other conceptual tables. All these differences must be taken into account when translating SMIv2 OBJECT-TYPE macro invocations to YANG. Invocations of the OBJECT-TYPE macro MUST be translated into YANG statements as detailed below.

7.1. Scalar and Columnar Object Translation Rules

SMIv2 OBJECT-TYPE macro invocations defining scalars or columnar objects with a MAX-ACCESS of "not-accessible", "read-only", "read-write", and "read-create" are translated to YANG leaf statements. Additionally, columnar objects with a MAX-ACCESS of "accessible-for-notify" are translated to YANG leaf statements if that columnar object is part of the INDEX clause of the table containing that columnar object. The name of the leaf is the name associated with the SMIv2 OBJECT-TYPE macro invocation. SMIv2 OBJECT-TYPE macro invocations with a MAX-ACCESS of

"accessible-for-notify" are not translated to YANG data tree leafs but instead are translated into YANG notification leafs.

Leaf statements for scalar objects are created in a container representing the scalar's parent node in the OID tree. This container is named after the scalar's parent node in the OID tree and placed in the top-level container representing the SMIv2 module; see Section 4.1. In the rare case that the scalar's parent node has multiple names, the automatic translation MUST fail with an error, and the name clash needs to be investigated and fixed manually. In case a previous revision of the SMIv2 module did not have an ambiguity, then the name used by the previous revision MUST be used. The leaf statements representing columnar objects are created in the list representing a conceptual row; see Section 7.3.

- The SMIv2 SYNTAX clause is mapped to the YANG type statement. SMIv2 range restrictions are mapped to YANG range statements, while SMIv2 length restrictions are mapped to YANG length statements. SMIv2 INTEGER enumerations are mapped to YANG enum/ value statements. SMIv2 BITS are mapped to YANG bit/position statements. For OCTET STRING types that are mapped to a YANG string base type (see Section 2), the length specified in the YANG length statement must be consistent with the stringified representation of values. If an implementation is unable to derive proper length restrictions, then the YANG length statement MUST be omitted.

- The SMIv2 UNITS clause is mapped to the YANG units statement.

- The SMIv2 MAX-ACCESS is translated into an smiv2:max-access statement. Refer to the YANG extension defined in Section 10.

- The SMIv2 STATUS clause is mapped to the YANG status statement. The generation of the YANG status statement is skipped if the value of the STATUS clause is current.

- The SMIv2 DESCRIPTION clause is mapped to the YANG description statement.

- The SMIv2 REFERENCE clause is mapped to the YANG reference statement.

- The SMIv2 DEFVAL clause is mapped to an smiv2:defval statement. Refer to the YANG extension defined in Section 10.

- The value of the SMIv2 OBJECT-TYPE macro invocation is translated into an smiv2:oid statement. Refer to the YANG extension defined in Section 10.

This translation assumes that labels of named numbers and named bits do not change when an SMIv2 module is revised. This is consistent with the clarification of the SMIv2 module revision rules in Section 4.9 of [RFC4181].

7.2. Example: ifNumber and ifIndex of the IF-MIB

The translations of the ifNumber scalar object and the ifIndex columnar object of the IF-MIB [RFC2863] are shown below. Since ifNumber is a scalar object in the interfaces branch of the IF-MIB, the YANG leaf ifNumber will be placed in a YANG container called interfaces, which is registered in the top-level container IF-MIB.

```
  leaf ifNumber {
    type int32;
    description
     "The number of network interfaces (regardless of their
      current state) present on this system.";
    smiv2:max-access "read-only";
    smiv2:oid "1.3.6.1.2.1.2.1";
  }

  leaf ifIndex {
    type if-mib:InterfaceIndex;
    description
     "A unique value, greater than zero, for each interface.  It
      is recommended that values are assigned contiguously
      starting from 1.  The value for each interface sub-layer
      must remain constant at least from one re-initialization of
      the entity's network management system to the next re-
      initialization.";
    smiv2:max-access "read-only";
    smiv2:oid "1.3.6.1.2.1.2.2.1.1";
  }
```

7.3. Non-Augmenting Conceptual Table Translation Rules

An OBJECT-TYPE macro invocation defining a non-augmenting conceptual table is translated to a YANG container statement using the name of the OBJECT-TYPE macro invocation. This container is created in the top-level container representing the SMIv2 module. The clauses of the macro are translated as follows:

- The SMIv2 SYNTAX clause is ignored
- The SMIv2 UNITS clause is ignored.
- The SMIv2 MAX-ACCESS clause is ignored.

- The SMIv2 STATUS clause is mapped to the YANG status statement. The generation of the YANG status statement is skipped if the value of the STATUS clause is current.

- The SMIv2 DESCRIPTION clause is mapped to the YANG description statement.

- The SMIv2 REFERENCE clause is mapped to the YANG reference statement.

- The value of the SMIv2 OBJECT-TYPE macro invocation is translated into an smiv2:oid statement. Refer to the YANG extension defined in Section 10.

An OBJECT-TYPE macro invocation defining a conceptual row is translated to a YANG list statement. It is contained in the YANG container representing the conceptual table. The generated list uses the name of the row OBJECT-TYPE macro invocation. The clauses of the OBJECT-TYPE macro are translated as follows:

- The SMIv2 SYNTAX clause is ignored.

- The SMIv2 UNITS clause is ignored.

- The SMIv2 MAX-ACCESS clause is ignored.

- The SMIv2 STATUS clause is mapped to the YANG status statement. The generation of the YANG status statement is skipped if the value of the STATUS clause is current.

- The SMIv2 DESCRIPTION clause is mapped to the YANG description statement.

- The SMIv2 REFERENCE clause is mapped to the YANG reference statement.

- The SMIv2 INDEX clause is mapped to the YANG key clause listing the columnar objects forming the key of the YANG list. If the same object appears more than once in the INDEX clause, append '_<n>' to the duplicate object name(s) where '<n>' counts the occurrences of the object in the INDEX clause, starting from 2. Additional leaf statements must be created to define the leafs introduced.

- If the SMIv2 INDEX clause contains the IMPLIED keyword, then an smiv2:implied statement is generated to record the name of the object preceded by the IMPLIED keyword. Refer to the YANG extension defined in Section 10.

- The value of the SMIv2 OBJECT-TYPE macro invocation is translated into an smiv2:oid statement. Refer to the YANG extension defined in Section 10.

Within the list statement, YANG leaf statements are created for columnar objects as described in Section 7.1. For objects listed in the SMIv2 INDEX clause that are not part of the conceptual table itself, YANG leaf statements of type leafref pointing to the referenced definition are created.

7.4. Example: ifTable of the IF-MIB

The translation of the definition of the ifTable of the IF-MIB [RFC2863] is shown below.

```
  container ifTable {
    description
     "A list of interface entries.  The number of entries is
      given by the value of ifNumber.";
    smiv2:oid "1.3.6.1.2.1.2.2";

    list ifEntry {
      key "ifIndex";
      description
       "An entry containing management information applicable to a
        particular interface.";
      smiv2:oid "1.3.6.1.2.1.2.2.1";

      leaf ifIndex {
        type if-mib:InterfaceIndex;
        description
         "A unique value, greater than zero, for each interface.  It
          is recommended that values are assigned contiguously
          starting from 1.  The value for each interface sub-layer
          must remain constant at least from one re-initialization of
          the entity's network management system to the next re-
          initialization.";
        smiv2:max-access "read-only";
        smiv2:oid "1.3.6.1.2.1.2.2.1.1";
      }

      // ...
    }
  }
```

7.5. Example: ifRcvAddressTable of the IF-MIB

The translation of the definition of the ifRcvAddressTable of the IF-MIB [RFC2863] is shown below.

```
  container ifRcvAddressTable {
    description
     "This table contains an entry for each address (broadcast,
      multicast, or uni-cast) for which the system will receive
      packets/frames on a particular interface, except as follows:

      - for an interface operating in promiscuous mode, entries are
        only required for those addresses for which the system would
        receive frames were it not operating in promiscuous mode.

      - for 802.5 functional addresses, only one entry is required,
        for the address which has the functional address bit ANDed
        with the bit mask of all functional addresses for which the
        interface will accept frames.

      A system is normally able to use any unicast address which
      corresponds to an entry in this table as a source address.";
    smiv2:oid "1.3.6.1.2.1.31.1.4";

    list ifRcvAddressEntry {
      key "ifIndex ifRcvAddressAddress";
      description
       "A list of objects identifying an address for which the
        system will accept packets/frames on the particular
        interface identified by the index value ifIndex.";
      smiv2:oid "1.3.6.1.2.1.31.1.4.1";

      leaf ifIndex {
        type leafref {
          path "/if-mib:IF-MIB/if-mib:ifTable" +
               "/if-mib:ifEntry/if-mib:ifIndex";
        }
      }

      leaf ifRcvAddressAddress {
        type yang:phys-address;
        description
         "An address for which the system will accept packets/frames
          on this entry's interface.";
        smiv2:max-access "not-accessible";
        smiv2:oid "1.3.6.1.2.1.31.1.4.1.1";
      }
```

```
        // ...
      }
    }
```

7.6. Example: alHostTable of the RMON2-MIB

The translation of the definition of the alHostTable of the RMON2-MIB [RFC4502] is shown below.

```
container alHostTable {
  description
   "A collection of statistics for a particular protocol from a
    particular network address that has been discovered on an
    interface of this device.

    The probe will populate this table for all protocols in the
    protocol directory table whose value of
    protocolDirHostConfig is equal to supportedOn(3), and
    will delete any entries whose protocolDirEntry is deleted or
    has a protocolDirHostConfig value of supportedOff(2).

    The probe will add to this table all addresses
    seen as the source or destination address in all packets with
    no MAC errors and will increment octet and packet counts in
    the table for all packets with no MAC errors.  Further,
    entries will only be added to this table if their address
    exists in the nlHostTable and will be deleted from this table
    if their address is deleted from the nlHostTable.";
  smiv2:oid "1.3.6.1.2.1.16.16.1";

  list alHostEntry {
    key "hlHostControlIndex alHostTimeMark protocolDirLocalIndex "
      + "nlHostAddress protocolDirLocalIndex_2";
    description
     "A conceptual row in the alHostTable.

      The hlHostControlIndex value in the index identifies the
      hlHostControlEntry on whose behalf this entry was created.
      The first protocolDirLocalIndex value in the index identifies
      the network-layer protocol of the address.
      The nlHostAddress value in the index identifies the network-
      layer address of this entry.
      The second protocolDirLocalIndex value in the index identifies
      the protocol that is counted by this entry.

      An example of the indexing in this entry is
      alHostOutPkts.1.783495.18.4.128.2.6.6.34.
```

```
          Note that some combinations of index values may result in an
          index that exceeds 128 sub-identifiers in length, which exceeds
          the maximum for the SNMP protocol.  Implementations should take
          care to avoid such combinations.";
        smiv2:oid "1.3.6.1.2.1.16.16.1.1";

        // ...

        leaf protocolDirLocalIndex {
          type leafref {
            path "/rmon2-mib:RMON2-MIB/"
               + "rmon2-mib:protocolDirTable/"
               + "rmon2-mib:protocolDirEntry/"
               + "rmon2-mib:protocolDirLocalIndex";
          }
        }

        // ...

        leaf protocolDirLocalIndex_2 {
          type leafref {
            path "/rmon2-mib:RMON2-MIB/"
               + "rmon2-mib:protocolDirTable/"
               + "rmon2-mib:protocolDirEntry/"
               + "rmon2-mib:protocolDirLocalIndex";
          }
        }

        // ...
      }
    }
```

7.7. Augmenting Conceptual Tables Translation Rules

An OBJECT-TYPE macro invocation defining an augmenting conceptual table is translated to a YANG smiv2:alias statement. Refer to the YANG extension defined in Section 10. The clauses of the macro are translated as follows:

- The SMIv2 SYNTAX clause is ignored.
- The SMIv2 UNITS clause is ignored.
- The SMIv2 MAX-ACCESS clause is ignored.
- The SMIv2 STATUS clause is mapped to the YANG status statement. The generation of the YANG status statement is skipped if the value of the STATUS clause is current.

- The SMIv2 DESCRIPTION clause is mapped to the YANG description statement.
- The SMIv2 REFERENCE clause is mapped to the YANG reference statement.
- The value of the SMIv2 OBJECT-TYPE macro invocation is translated into an smiv2:oid statement. Refer to the YANG extension defined in Section 10.

An OBJECT-TYPE macro invocation defining a conceptual row augmentation is translated to a YANG smiv2:alias statement and a YANG augment statement using the path to the augmented table as its argument. The clauses of the OBJECT-TYPE macro are translated as follows:

- The SMIv2 SYNTAX clause is ignored.
- The SMIv2 UNITS clause is ignored.
- The SMIv2 MAX-ACCESS clause is ignored.
- The SMIv2 STATUS clause is mapped to the YANG status statement. The generation of the YANG status statement is skipped if the value of the STATUS clause is current.
- The SMIv2 DESCRIPTION clause is mapped to the YANG description statement.
- The SMIv2 REFERENCE clause is mapped to the YANG reference statement.
- The value of the SMIv2 OBJECT-TYPE macro invocation is translated into an smiv2:oid statement. Refer to the YANG extension defined in Section 10.

Within the augment statement, YANG leaf statements are created as described in Section 7.1.

7.8. Example: ifXTable of the IF-MIB

The translation of the definition of the ifXTable of the IF-MIB [RFC2863] is shown below.

```
  smiv2:alias "ifXTable" {
    description
     "A list of interface entries.  The number of entries is
      given by the value of ifNumber.  This table contains
      additional objects for the interface table.";
    smiv2:oid "1.3.6.1.2.1.31.1.1";
  }

  smiv2:alias "ifXEntry" {
    description
     "An entry containing additional management information
      applicable to a particular interface.";
    smiv2:oid "1.3.6.1.2.1.31.1.1.1";
  }

  augment "/if-mib:IF-MIB/if-mib:ifTable/if-mib:ifEntry" {
    description
     "An entry containing additional management information
      applicable to a particular interface.";
    smiv2:oid "1.3.6.1.2.1.31.1.1.1";

    leaf ifName {
      type snmpv2-tc:DisplayString;
      description
       "The textual name of the interface.  The value of this
        object should be the name of the interface as assigned by
        the local device and should be suitable for use in commands
        entered at the device's 'console'.  This might be a text
        name, such as 'le0' or a simple port number, such as '1',
        depending on the interface naming syntax of the device.  If
        several entries in the ifTable together represent a single
        interface as named by the device, then each will have the
        same value of ifName.  Note that for an agent which responds
        to SNMP queries concerning an interface on some other
        (proxied) device, then the value of ifName for such an
        interface is the proxied device's local name for it.

        If there is no local name, or this object is otherwise not
        applicable, then this object contains a zero-length string.";
      smiv2:max-access "read-only";
      smiv2:oid "1.3.6.1.2.1.31.1.1.1.1";
    }
```

```
      // ...
    }
```

8. Translation of the OBJECT-IDENTITY Macro

The SMIv2 uses invocations of the OBJECT-IDENTITY macro to define information about an OBJECT IDENTIFIER assignment. Invocations of the OBJECT-IDENTITY macro MUST be translated into YANG identity statements as detailed below.

8.1. OBJECT-IDENTITY Translation Rules

The name of the OBJECT-IDENTITY macro invocation is used as the name of the generated identity statement. The generated identity statement uses the smiv2:object-identity defined in Section 10 as its base. The clauses of the SMIv2 OBJECT-IDENTITY macro are mapped to YANG statements as follows:

- The SMIv2 STATUS clause is mapped to the YANG status statement. The generation of the YANG status statement is skipped if the value of the STATUS clause is current.
- The SMIv2 DESCRIPTION clause is mapped to the YANG description statement.
- The SMIv2 REFERENCE clause is mapped to the YANG reference statement.
- The value of the SMIv2 OBJECT-IDENTITY macro invocation is translated into an smiv2:oid statement. Refer to the YANG extension defined in Section 10.

8.2. Example: diffServTBParamSimpleTokenBucket of the DIFFSERV-MIB

The translation of the diffServTBParamSimpleTokenBucket of the DIFFSERV-MIB [RFC3289] is shown below. (Please note that the description should refer to RFC 3290, Section 5.1.3.)

```
  identity diffServTBParamSimpleTokenBucket {
    base "smiv2:object-identity";
    description
     "Two Parameter Token Bucket Meter as described in the Informal
      Differentiated Services Model section 5.2.3.";
    smiv2:oid "1.3.6.1.2.1.97.3.1.1";
  }
```

9. Translation of the NOTIFICATION-TYPE Macro

SMIv2 provides the NOTIFICATION-TYPE macro to define event notifications. YANG provides the notification statement for the same purpose. Invocations of the NOTIFICATION-TYPE macro MUST be translated into YANG notification statements as detailed below.

9.1. NOTIFICATION-TYPE Translation Rules

The name of the NOTIFICATION-TYPE macro invocation is used as the name of the generated notification statement. The clauses of the NOTIFICATION-TYPE macro are mapped to YANG statements embedded in the notification statement as follows.

- The SMIv2 OBJECTS clause is mapped to a sequence of YANG containers. For each object listed in the OBJECTS clause value, a YANG container statement is generated. The name of this container is the string "object-<n>", where <n> is the position of the object in the value of the OBJECTS clause (first element has position 1). If the current object belongs to a conceptual table, then a sequence of leaf statements is generated for each INDEX object of the conceptual table. These leafs are named after the INDEX objects and of type leafref. Finally, a leaf statement is generated named after the current object. If the current object has a MAX-ACCESS of "read-only", "read-write", or "read-create", then the generated leaf is of type leafref. Otherwise, if the current object has a MAX-ACCESS of "accessible-for-notify", then a leaf is generated, following the steps in Section 7.1.

- The SMIv2 STATUS clause is mapped to the YANG status statement. The generation of the YANG status statement is skipped if the value of the STATUS clause is current.

- The SMIv2 DESCRIPTION clause is mapped to the YANG description statement.

- The SMIv2 REFERENCE clause is mapped to the YANG reference statement.

- The value of the SMIv2 NOTIFICATION-TYPE macro invocation is translated into an smiv2:oid statement. Refer to the YANG extension defined in Section 10.

9.2. Example: linkDown NOTIFICATION-TYPE of IF-MIB

The translation of the linkDown notification of the IF-MIB [RFC2863] is shown below.

```
  notification linkDown {
    description
     "A linkDown trap signifies that the SNMP entity, acting in
      an agent role, has detected that the ifOperStatus object for
      one of its communication links is about to enter the down
      state from some other state (but not from the notPresent
      state).  This other state is indicated by the included value
      of ifOperStatus.";
    smiv2:oid "1.3.6.1.6.3.1.1.5.3";

    container object-1 {
      leaf ifIndex {
        type leafref {
          path "/if-mib:IF-MIB/if-mib:ifTable" +
               "/if-mib:ifEntry/if-mib:ifIndex";
        }
      }
    }

    container object-2 {
      leaf ifIndex {
        type leafref {
          path "/if-mib:IF-MIB/if-mib:ifTable" +
               "/if-mib:ifEntry/if-mib:ifIndex";
        }
      }
      leaf ifAdminStatus {
        type leafref {
          path "/if-mib:IF-MIB/if-mib:ifTable" +
               "/if-mib:ifEntry/if-mib:ifAdminStatus";
        }
      }
    }

    container object-3 {
      leaf ifIndex {
        type leafref {
          path "/if-mib:IF-MIB/if-mib:ifTable" +
               "/if-mib:ifEntry/if-mib:ifIndex";
        }
      }
      leaf ifOperStatus {
        type leafref {
```

```
            path "/if-mib:IF-MIB/if-mib:ifTable" +
                 "/if-mib:ifEntry/if-mib:ifOperStatus";
          }
        }
      }
    }
```

10. YANG Language Extension Definition

This section defines some YANG extension statements that can be used to capture some information present in SMIv2 modules that is not translated into core YANG statements. The YANG module references [RFC2578] and [RFC2579].

<CODE BEGINS> file "ietf-yang-smiv2@2012-06-22.yang"

```
 module ietf-yang-smiv2 {

   namespace "urn:ietf:params:xml:ns:yang:ietf-yang-smiv2";
   prefix "smiv2";

   organization
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

   contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     WG Chair: David Kessens
               <mailto:david.kessens@nsn.com>

     WG Chair: Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>

     Editor:   Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>";

   description
    "This module defines YANG extensions that are used to translate
     SMIv2 concepts into YANG.

     Copyright (c) 2012 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
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC 6643; see
    the RFC itself for full legal notices.";

  revision 2012-06-22 {
    description
     "Initial revision.";
    reference
     "RFC 6643: Translation of Structure of Management Information
      Version 2 (SMIv2) MIB Modules to YANG Modules";
  }

  identity object-identity {
    description
     "Base identity for all SMIv2 OBJECT-IDENTITYs.";
  }

  typedef opaque {
    type binary;
    description
     "The Opaque type supports the capability to pass arbitrary ASN.1
      syntax.  A value is encoded using the ASN.1 Basic Encoding Rules
      into a string of octets.  This, in turn, is encoded as an OCTET
      STRING, in effect 'double-wrapping' the original ASN.1 value.

      In the value set and its semantics, this type is equivalent to
      the Opaque type of the SMIv2.  This type exists in the SMIv2
      solely for backward-compatibility reasons and this is also
      true for this YANG data type.";
    reference
     "RFC 2578: Structure of Management Information Version 2 (SMIv2)";
  }

  extension display-hint {
    argument "format";
    description
     "The display-hint statement takes as an argument the DISPLAY-HINT
      assigned to an SMIv2 textual convention.";
    reference
     "RFC 2579: Textual Conventions for SMIv2";
  }
```

```
   extension max-access {
     argument "access";
     description
      "The max-access statement takes as an argument the MAX-ACCESS
       assigned to an SMIv2 object definition.

       The MAX-ACCESS value is SMIv2 specific and has no impact on
       the access provided to YANG objects through protocols such
       as NETCONF.";
     reference
      "RFC 2578: Structure of Management Information Version 2 (SMIv2)";
   }

   extension defval {
     argument "value";
     description
      "The defval statement takes as an argument a default value
       defined by an SMIv2 DEFVAL clause.  Note that the value is in
       the SMIv2 value space defined by the SMIv2 syntax of the
       corresponding object and not in the YANG value space
       defined by the corresponding YANG data type.";
     reference
      "RFC 2578: Structure of Management Information Version 2 (SMIv2)";
   }

   extension implied {
     argument "index";
     description
      "If an SMIv2 INDEX object is preceded by the IMPLIED keyword, then
       the implied statement is present in the YANG module and takes as
       an argument the name of the IMPLIED index object.";
     reference
      "RFC 2578: Structure of Management Information Version 2 (SMIv2)";
    }

   extension alias {
     argument "descriptor";
     description
      "The alias statement introduces an SMIv2 descriptor.  The body of
       the alias statement is expected to contain an oid statement that
       provides the numeric OID associated with the descriptor.";
     reference
      "RFC 2578: Structure of Management Information Version 2 (SMIv2)";
   }
```

```
  extension oid {
    argument "value";
    description
     "The oid statement takes as an argument the object identifier
      assigned to an SMIv2 definition.  The object identifier value
      is written in decimal dotted notation.";
    reference
     "RFC 2578: Structure of Management Information Version 2 (SMIv2)";
  }

  extension subid {
    argument "value";
    description
     "The subid statement takes as an argument the last sub-identifier
      of the object identifier assigned to an SMIv2 definition.  The
      sub-identifier value is a single positive decimal natural number.
      The subid statement may not be used as a substatement to any
      top-level node in a YANG document.  The subid substatement may
      be used only as a substatement to a node having a parent node
      defined with either an smiv2:oid or smiv2:subid substatement.";
    reference
     "RFC 2578: Structure of Management Information Version 2 (SMIv2)";
  }

}

  <CODE ENDS>
```

11. Implementing Configuration Data Nodes

The result of the translation of SMIv2 MIB modules into YANG modules, even if SMIv2 objects are read-write or read-create, consists of read-only (config false) YANG objects. One reason is that the persistency models of the underlying protocols, SNMP and NETCONF, are quite different. With SNMP, the persistency of a writable object depends either on the object definition itself (i.e., the text in the DESCRIPTION clause) or the persistency properties of the conceptual row it is part of, sometimes controlled via a columnar object using the StorageType textual convention. With NETCONF, the persistency of configuration objects is determined by the properties of the underlying datastore. Furthermore, NETCONF as defined in [RFC6241] does not provide a standard operation to modify operational state. The <edit-config> and <copy-config> operations only manipulate configuration data. As a consequence of these considerations, it is not possible to generate YANG configuration data nodes from SMIv2 definitions in an automated way.

However, for selected SMIv2 objects where the SNMP and NETCONF persistency semantics are consistent, implementations may choose to implement some YANG data nodes generated from SMIv2 definitions as configuration data nodes. Such a deviation from the generated read-only YANG module should be formally documented in the form of a separate YANG module that uses YANG deviation statements to change the config property of the data nodes implemented as configuration data nodes from false to true. Deviations that change the config false property to true without any other changes to the semantics of the data node do not affect the compliance with the YANG module generated from an SMIv2 module.

11.1. Example: addressMapControlTable of RMON2-MIB

The following example demonstrates how certain columnar objects of the addressMapControlTable of the RMON2-MIB [RFC4502] can be turned into YANG configuration data nodes. Note that YANG deviations affect the property of the target node only and are not inherited downwards.

```
  module acme-RMON2-MIB-deviations {

    namespace "http://acme.example.com/RMON2-MIB-deviations";
    prefix "acme-rmon2-devs";

    import RMON2-MIB {
      prefix "rmon2-mib";
      revision-date 2006-05-02;
    }

    revision 2012-01-11 {
      description
        "First version.";
    }

    deviation "/rmon2-mib:RMON2-MIB" {
      deviate replace {
        config true;
      }
    }

    deviation "/rmon2-mib:RMON2-MIB/"
            + "rmon2-mib:addressMapControlTable" {
      deviate replace {
        config true;
      }
    }
```

```
      deviation "/rmon2-mib:RMON2-MIB/"
              + "rmon2-mib:addressMapControlTable/"
              + "rmon2-mib:addressMapControlEntry" {
        deviate replace {
          config true;
        }
      }

      deviation "/rmon2-mib:RMON2-MIB/"
              + "rmon2-mib:addressMapControlTable/"
              + "rmon2-mib:addressMapControlEntry/"
              + "rmon2-mib:addressMapControlIndex" {
        deviate replace {
          config true;
        }
      }

      deviation "/rmon2-mib:RMON2-MIB/"
              + "rmon2-mib:addressMapControlTable/"
              + "rmon2-mib:addressMapControlEntry/"
              + "rmon2-mib:addressMapControlDataSource" {
        deviate replace {
          config true;
        }
      }

      deviation "/rmon2-mib:RMON2-MIB/"
              + "rmon2-mib:addressMapControlTable/"
              + "rmon2-mib:addressMapControlEntry/"
              + "rmon2-mib:addressMapControlOwner" {
        deviate replace {
          config true;
        }
      }
    }
```

A NETCONF server that implements the RMON2-MIB module with these deviations would advertise the following capabilities in its <hello> message (where whitespace has been added for readability):

```
    <capability>
      urn:ietf:params:xml:ns:yang:smiv2:RMON2-MIB?
        module=RMON2-MIB&amp;amp;
        revision=2006-05-02&amp;amp;
        deviations=acme-RMON2-MIB-deviations
    </capability>
    <capability>
      http://acme.example.com/RMON2-MIB-deviations?
        module=acme-RMON2-MIB-deviations&amp;amp;
        revision=2012-01-11
    </capability>
```

12. IANA Considerations

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registrations have been made.

URI: urn:ietf:params:xml:ns:yang:ietf-yang-smiv2

Registrant Contact: The NETMOD WG of the IETF.

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

URI: urn:ietf:params:xml:ns:yang:smiv2

Registrant Contact: The NETMOD WG of the IETF.

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

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
Name:         ietf-yang-smiv2
Namespace:    urn:ietf:params:xml:ns:yang:ietf-yang-smiv2
Prefix:       smiv2
Reference:    RFC 6643
```

13. Security Considerations

This document defines a translation of SMIv2 MIB modules into YANG modules, enabling read-only (config false) access to data objects defined in SMIv2 MIB modules via NETCONF. The translation itself has no security impact on the Internet.

Users of YANG data models generated from SMIv2 data models that have been published in the RFC series are advised to consult the security considerations of the respective RFCs. The security considerations of RFCs containing SMIv2 data models explain which objects are sensitive and important to protect. NETCONF users are encouraged to make use of the NETCONF access control model [RFC6536], which allows the specification of access control rules to protect potentially sensitive information.

14. Acknowledgements

The author wishes to thank the following individuals for providing helpful comments on various draft versions of this document: Andy Bierman, Benoit Claise, Martin Bjorklund, Leif Johansson, David Reid, Dan Romascanu, and David Spakes.

15. References

15.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

15.2. Informative References

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[RFC3289] Baker, F., Chan, K., and A. Smith, "Management Information Base for the Differentiated Services Architecture", RFC 3289, May 2002.

[RFC3584] Frye, R., Levi, D., Routhier, S., and B. Wijnen, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework", RFC 3584, August 2003.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC4181] Heard, C., "Guidelines for Authors and Reviewers of MIB Documents", BCP 111, RFC 4181, September 2005.

[RFC4502] Waldbusser, S., "Remote Network Monitoring Management Information Base Version 2", RFC 4502, May 2006.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6536] Bierman, A., Ed. and M. Bjorklund, Ed., "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Appendix A. Mapping of Well-Known Types (Normative)

This normative appendix describes the mapping of SMIv2 types to YANG types. The mapping is fully consistent with Tables 1 and 2 of [RFC6021].

```
SMIv2 Module: SNMPv2-SMI
SMIv2 Type:   INTEGER       (used as an enumeration)
YANG Type:    enumeration

SMIv2 Module: SNMPv2-SMI
SMIv2 Type:   INTEGER       (used as a numeric type)
YANG Type:    int32

SMIv2 Module: SNMPv2-SMI
SMIv2 Type:   Integer32
YANG Type:    int32

SMIv2 Module: SNMPv2-SMI
SMIv2 Type:   OCTET STRING  (used as a binary string)
YANG Type:    binary

SMIv2 Module: SNMPv2-SMI
SMIv2 Type:   OCTET STRING  (used to hold UTF-8 or ASCII characters)
YANG Type:    string

SMIv2 Module: SNMPv2-SMI
SMIv2 Type:   OBJECT IDENTIFIER
YANG Module:  ietf-yang-types
YANG Type:    object-identifier-128

SMIv2 Module: SNMPv2-SMI
SMIv2 Type:   BITS
YANG Type:    bits

SMIv2 Module: SNMPv2-SMI
SMIv2 Type:   IpAddress
YANG Module:  ietf-inet-types
YANG Type:    ipv4-address

SMIv2 Module: SNMPv2-SMI
SMIv2 Type:   Counter32
YANG Module:  ietf-yang-types
YANG Type:    counter32
```

```
   SMIv2 Module: SNMPv2-SMI
   SMIv2 Type:   Gauge32
   YANG Module:  ietf-yang-types
   YANG Type:    gauge32

   SMIv2 Module: SNMPv2-SMI
   SMIv2 Type:   TimeTicks
   YANG Module:  ietf-yang-types
   YANG Type:    timeticks

   SMIv2 Module: SNMPv2-SMI
   SMIv2 Type:   Counter64
   YANG Module:  ietf-yang-types
   YANG Type:    counter64

   SMIv2 Module: SNMPv2-SMI
   SMIv2 Type:   Unsigned32
   YANG Type:    uint32

   SMIv2 Module: SNMPv2-SMI
   SMIv2 Type:   Opaque
   YANG Module:  ietf-yang-smiv2
   YANG Type:    opaque

   SMIv2 Module: SNMPv2-TC
   SMIv2 Type:   PhysAddress
   YANG Module:  ietf-yang-types
   YANG Type:    phys-address

   SMIv2 Module: SNMPv2-TC
   SMIv2 Type:   MacAddress
   YANG Module:  ietf-yang-types
   YANG Type:    mac-address

   SMIv2 Module: SNMPv2-TC
   SMIv2 Type:   TruthValue
   YANG Type:    boolean

   SMIv2 Module: SNMPv2-TC
   SMIv2 Type:   TimeStamp
   YANG Module:  ietf-yang-types
   YANG Type:    timestamp

   SMIv2 Module: RMON2-MIB
   SMIv2 Type:   ZeroBasedCounter32
   YANG Module:  ietf-yang-types
   YANG Type:    zero-based-counter32
```

```
SMIv2 Module: HCNUM-TC
SMIv2 Type:   ZeroBasedCounter64
YANG Module:  ietf-yang-types
YANG Type:    zero-based-counter64

SMIv2 Module: HCNUM-TC
SMIv2 Type:   CounterBasedGauge64
YANG Module:  ietf-yang-types
YANG Type:    gauge64

SMIv2 Module: INET-ADDRESS-MIB
SMIv2 Type:   InetAutonomousSystemNumber
YANG Module:  ietf-inet-types
YANG Type:    as-number

SMIv2 Module: INET-ADDRESS-MIB
SMIv2 Type:   InetVersion
YANG Module:  ietf-inet-types
YANG Type:    ip-version

SMIv2 Module: INET-ADDRESS-MIB
SMIv2 Type:   InetPortNumber
YANG Module:  ietf-inet-types
YANG Type:    port-number

SMIv2 Module: DIFFSERV-DSCP-TC
SMIv2 Type:   Dscp
YANG Module:  ietf-inet-types
YANG Type:    dscp

SMIv2 Module: IPV6-FLOW-LABEL-MIB
SMIv2 Type:   IPv6FlowLabel
YANG Module:  ietf-inet-types
YANG Type:    ipv6-flow-label

SMIv2 Module: URI-TC-MIB
SMIv2 Type:   Uri
YANG Module:  ietf-inet-types
YANG Type:    uri
```

Appendix B. Module Prefix Generation (Informative)

This section describes an algorithm to generate module prefixes to be used in the import statements. The input of the prefix generation algorithm is a set of prefixes (usually derived from imported module names) and a specific module name to be converted into a prefix. The algorithm described below produces a prefix for the given module name that is unique within the set of prefixes.

YANG Module	Prefix
ietf-yang-types	yang
ietf-inet-types	inet
ietf-yang-smiv2	smiv2

Table 1: Special Prefixes For Well-Known YANG Modules

- First, some predefined translations mapping well-known YANG modules to short prefixes are tried (see Table 1). If a fixed translation rule exists and leads to a conflict-free prefix, then the fixed translation is used.

- Otherwise, prefixes are generated by tokenizing a YANG module name, using hyphens as token separators. The tokens derived from the module name are converted to lowercase characters. The prefix then becomes the shortest sequence of tokens concatenated using hyphens as separators, which includes at least two tokens and which is unique among all prefixes used in the YANG module.

In the worst case, the prefix derived from an SMIv2 module name becomes the SMIv2 module name translated to lowercase. But on average, much shorter prefixes are generated.

Author's Address

Juergen Schoenwaelder
Jacobs University

EMail: j.schoenwaelder@jacobs-university.de