Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 15, 2011

J. Schoenwaelder
Jacobs University

March 14, 2011

Translation of SMIv2 MIB Modules to YANG Modules
draft-schoenw-netmod-smi-yang-02

Abstract

YANG is a data modeling language used to model configuration and state data manipulated by the NETCONF protocol, 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 SNMP protocol. This document defines a translation of SMIv2 MIB modules into YANG modules, enabling read-only access to data objects defined in SMIv2 MIB modules via NETCONF.

Table of Contents

1. Introduction 4
2. Mapping of Special Types 5
3. Module Prefix Generation 6
4. Translation of SMIv2 Modules and SMIv2 IMPORT Clauses 7
4.1. Example: IMPORTS of IF-MIB 8
5. Translation of the MODULE-IDENTITY Macro 9
5.1. MODULE-IDENTITY Translation Rules 9
5.2. Example: MODULE-IDENTITY of IF-MIB 9
6. Translation of the TEXTUAL-CONVENTION Macro 11
6.1. TEXTUAL-CONVENTION Translation Rules 11
6.2. Example: OwnerString and InterfaceIndex of IF-MIB 11
6.3. Example: IfDirection of the DIFFSERV-MIB 12
7. Translation of OBJECT IDENTIFIER Assignments 13
7.1. Object Identifier Assignment Translation Rules 13
7.2. Example: OBJECT IDENTIFIER Assignments of the IF-MIB . . . 13
8. Translation of the OBJECT-TYPE Macro 14
8.1. Scalar and Columnar Object Translation Rules 14
8.2. Example: ifNumber and ifIndex of the IF-MIB 14
8.3. Non-Augmenting Conceptual Table Translation Rules 15
8.4. Example: ifTable of the IF-MIB 16
8.5. Example: ifRcvAddressTable of the IF-MIB 16
8.6. Augmenting Conceptual Tables Translation Rules 18
8.7. Example: ifXTable of the IF-MIB 18
9. Translation of the OBJECT-IDENTITY Macro 20
9.1. OBJECT-IDENTITY Translation Rules 20
9.2. Example: diffServTBParamSimpleTokenBucket of the DIFFSERV-MIB 20
10. Translation of the NOTIFICATION-TYPE Macro 21
10.1. NOTIFICATION-TYPE Translation Rules 21
10.2. Example: linkDown NOTIFICATION-TYPE of IF-MIB 21
11. YANG Language Extension Definition 24
12. IANA Considerations 26
13. Security Considerations 27
14. References 28
14.1. Normative References 28
14.2. Informative References 28
Appendix A. Changes from 01 to 02 29
Appendix B. Changes from 00 to 01 30
Author's Address 31

1. Introduction

This document describes an translation of SMIv2 [RFC2578], [RFC2579], [RFC2580] MIB modules into YANG [RFC6020] modules, enabling read-only access to data objects defined in SMIv2 MIB modules via NETCONF. The mapping is illustrated by examples showing the translation of part of the IF-MIB [RFC2863] SMIv2 module and the DIFFSERV-MIB [RFC3289] SMIv2 module.

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 Special Types

The SMIv2 base types and some well known derived textual-conventions are mapped to YANG types according to Table 1. 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. Otherwise, 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.

Mapping of SMIv2 types to YANG types

SMIv2 Module	SMIv2 Type	YANG Module	YANG Type
SNMPv2-SMI	INTEGER		enumeration
SNMPv2-SMI	INTEGER		int32
SNMPv2-SMI	Integer32		int32
SNMPv2-SMI	OCTET STRING		binary
SNMPv2-SMI	OCTET STRING		string
SNMPv2-SMI	OBJECT IDENTIFIER	ietf-yang-types	object-identifier
SNMPv2-SMI	BITS		bits
SNMPv2-SMI	IpAddress	ietf-inet-types	ipv4-address
SNMPv2-SMI	Counter32	ietf-yang-types	counter32
SNMPv2-SMI	Gauge32	ietf-yang-types	gauge32
SNMPv2-SMI	TimeTicks	ietf-yang-types	timeticks
SNMPv2-SMI	Opaque		binary
SNMPv2-SMI	Counter64	ietf-yang-types	counter64
SNMPv2-SMI	Unsigned32		uint32
SNMPv2-TC	PhysAddress	ietf-yang-types	phys-address
SNMPv2-TC	MacAddress	ietf-yang-types	mac-address
SNMPv2-TC	TimeStamp	ietf-yang-types	timestamp

Table 1

The mappings shown in Table 1 may impact the imports of the generated YANG module since some SMIv2 types and textual-conventions map to YANG types defined in the ietf-yang-types and ietf-inet-types YANG modules [RFC6021]. Implementations must add any additional imports required by the type mapping.

3. Module Prefix Generation

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.

Special prefixes for well known SMIv2 and YANG modules

YANG / SMIv2 Module	Prefix
ietf-yang-types	yang
ietf-inet-types	inet
ietf-yang-smiv2	smiv2

Table 2

- First, some fixed translations mapping well known SMIv2 and YANG modules to short prefixes are tried (see Table 2). 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 an SMIv2 module name where hyphens are considered as token separators. The tokens derived with a module name are converted to lowercase characters. The prefix then becomes the shortest sequence of token concatenated using hyphens as separators, which includes at least two token 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 lower-case. But on average, much shorter prefixes are generated.

4. Translation of SMIv2 Modules and SMIv2 IMPORT Clauses

SMIv2 modules are mapped to corresponding YANG modules. The YANG module name is the same as the SMIv2 module name.

The YANG namespace is constructed out of a constant prefix followed by the SMIv2 module name. Since SMIv2 module names are unique, the resulting YANG namespace is unique. The registered prefix is urn:ietf:params:xml:ns:yang:smiv2:, see the IANA considerations section.

The YANG prefix is derived from the SMIv2 module name using the module prefix generation algorithm described in Section 3. 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 Section 3 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.

SMIv2 imports that are ignored in YANG

SMIv2 Module	SMIv2 Symbol
SNMPv2-SMI	MODULE-IDENTITY
SNMPv2-SMI	OBJECT-IDENTITY
SNMPv2-SMI	OBJECT-TYPE
SNMPv2-SMI	NOTIFICATION-TYPE
SNMPv2-SMI	mib-2
SNMPv2-TC	TEXTUAL-CONVENTION
SNMPv2-CONF	OBJECT-GROUP
SNMPv2-CONF	NOTIFICATION-GROUP
SNMPv2-CONF	MODULE-COMPLIANCE
SNMPv2-CONF	AGENT-CAPABILITIES
SNMPv2-MIB	snmpTraps
SNMPv2-SMI	all symbols
SNMPv2-CONF	all symbols

Table 3

In order to produce correct and complete YANG import statements, it is necessary to apply the following rules:

- Ignore all imports listed in Table 3. Note that the modules SNMPv2-SMI and SNMPv2-CONF are completely ignored since all definitions in these modules are translated by translation rules.

- Add any imports required by the type translations according to the type mapping table. This requires to consider all the types used in the translation unit.

The argument of the generated import statements are the untranslated SMIv2 module name. The import statement must contain a prefix statement. The prefixes are generated by applying the module prefix generation algorithm described in Section 3.

4.1. Example: IMPORTS of IF-MIB

The translation of the IF-MIB [RFC2863] leads to the YANG module frame and the import statements shown below. The prefix is the translation of the SMIv2 module name IF-MIB to lowercase (consisting of two token 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 "smiv2-tc"; }
  import ietf-yang-types     { prefix "yang"; }
  import ietf-yang-smiv2     { prefix "smiv2"; }
}
```

5. Translation of the MODULE-IDENTITY Macro

The clauses of the SMIv2 MODULE-IDENTITY macro are mapped to equivalent YANG statements.

5.1. 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 of contained in the SMIv2 REVISION. DESCRIPTION sub-clauses of REVISION clauses are mapped to corresponding description statement nested in revision clauses.
- The SMIv2 LAST-UPDATED is ignored if the associated date matches a REVISION clause. Otherwise, an additional revision statement is generated.
- The value of the invocation of an SMIv2 MODULE-IDENTITY macro is ignored.

While all proper SMIv2 modules must have a 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 out of 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, and revision statements.

5.2. 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.";
     }
```

6. 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 are translated into YANG typedef statements.

6.1. 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. [[TODO: Define a translation algorithm that is simple and produces correct and usable results for the majority of simple DISPLAY-HINTS?]]

- 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 and SMIv2 BITS are mapped to YANG enum / value and bit / position statements.

6.2. Example: OwnerString and InterfaceIndex of IF-MIB

The translation 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'.";
  }

  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.";
  }
```

6.3. 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.";
}
```

7. Translation of OBJECT IDENTIFIER Assignments

The mapping suppresses many structural OBJECT IDENTIFIER assignments that are typically used to organize the OBJECT IDENTIFIER tree.

7.1. Object Identifier Assignment Translation Rules

- Object identifier assignments through ASN.1 value assignments or through the invocation of a MODULE-IDENTITY clause are translated to YANG container statements.

- Top-level container must be marked as config false.

- Implementations MAY suppress the generation of YANG containers for object identifiers that only contain SMIv2 conformance definitions.

[[TODO: What do we do if multiple assignments exist for the same OID value?]]

7.2. Example: OBJECT IDENTIFIER Assignments of the IF-MIB

The translation of the OBJECT IDENTIFIER assignments and the value of the MODULE-IDENTITY clause of the IF-MIB [RFC2863] is shown below.

```
container interfaces {
  config false;
  // ...
}

container ifMIB {
  config false;
  container ifMIBObjects {
    // ...
  }

  container ifConformance {
    container ifGroups {
      // ...
    }
    container ifCompliances {
      // ...
    }
  }
}
```

8. 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 (zero or multiple instances within an SNMP context) within conceptual tables. A number of auxiliary objects define the index (key) of the table. Furthermore, conceptual tables can be augmented by other conceptual tables. All these differences must be taken into account when mapping SMIv2 OBJECT-TYPE macro invocations to YANG.

8.1. Scalar and Columnar Object Translation Rules

The SMIv2 OBJECT-TYPE macro invocations defining scalars or columnar objects are translated to YANG leaf statements. The name of the leaf is the name associated with the SMIv2 OBJECT-TYPE macro invocation.

- The SMIv2 SYNTAX clause is mapped to the YANG type clause. Embedded clauses are generates as described in Section 2.
- The SMIv2 UNITS clause is mapped to the YANG units statement.
- 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 ignored.

8.2. 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.

```
leaf ifNumber {
  type int32;
  description
   "The number of network interfaces (regardless of their
    current state) present on this system.";
}

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.";
}
```

8.3. Non-Augmenting Conceptual Table Translation Rules

An OBJECT-TYPE clause defining a non-augmenting conceptual table is translated to a YANG container statement using the name of the table OBJECT-TYPE clause. The OBJECT-TYPE clause representing a table row is translated to a YANG list statement using the name of the row OBJECT-TYPE clause. The rest of the clauses 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.
- The value of the SMIv2 OBJECT-TYPE macro invocation is ignored.

Within the list statement, YANG leaf statements are created for columnar objects as described above. 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.

8.4. 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.";

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

      // ...
    }
  }
```

8.5. 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.";

  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.";

    leaf ifIndex {
      type leafref {
        path "/if-mib:interfaces/if-mib:ifTable" +
             "/if-mib:ifEntry/if-mib:ifIndex";
      }
      description
       "[Automatically generated leaf for a foreign index.]";
    }

    leaf ifRcvAddressAddress {
      type yang:phys-address;
      description
       "An address for which the system will accept packets/frames
        on this entry's interface.";
    }

    // ...
  }
}
```

8.6. Augmenting Conceptual Tables Translation Rules

An OBJECT-TYPE clause defining an augmenting conceptual table is translated to a YANG container statement using the name of the table OBJECT-TYPE clause. The OBJECT-TYPE clause representing a table row is translated to a YANG augment statement using the path to the augmented table. The rest of the clauses 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 ignored.

Within the augment statement, YANG leaf nodes are created as described above.

8.7. Example: ifXTable of the IF-MIB

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

```
  container 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."

    augment "/if-mib:interfaces/if-mib:ifTable" +
            "/if-mib:ifEntry" {
      description
       "An entry containing additional management information
        applicable to a particular interface.";

      // ...
    }
```

```
    }
```

9. Translation of the OBJECT-IDENTITY Macro

Invocations of the OBJECT-IDENTITY macro are translated into YANG container statements.

9.1. OBJECT-IDENTITY Translation Rules

The name of the OBJECT-IDENTITY macro invocation is used as the name of the generated container statement. Any generated top-level container must be marked as config false. 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.

9.2. Example: diffServTBParamSimpleTokenBucket of the DIFFSERV-MIB

The translation of the diffServTBParamSimpleTokenBucket of the DIFFSERV-MIB [RFC3289] is shown below.

```
  container diffServTBParamSimpleTokenBucket {
    description
     "Two Parameter Token Bucket Meter as described in the Informal
      Differentiated Services Model section 5.2.3.";
  }
```

[[TODO: Should we in addition generate toplevel YANG identities so that definitions can be referenced from new YANG modules? See the example below (which assumes we provide an smiv2:object-identity base).]]

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

10. Translation of the NOTIFICATION-TYPE Macro

The SMIv2 provides the NOTIFICATION-TYPE macro to define notifications. YANG provides the notification statement for the same purpose.

10.1. 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 name of the notification and the name of the current concatenated by a hyphen. If the current object belongs a conceptual table, then a sequence of leaf statements is generated for each INDEX of the SMIv2 conceptual table. Next, a leaf statement is generated for the current object. All container leafs are marked as config false.

- 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 ignored.

10.2. 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.";

      container linkDown-ifIndex {
        config false;
        leaf ifIndex {
          type leafref {
            path "/if-mib:interfaces/if-mib:ifTable" +
                 "/if-mib:ifEntry/if-mib:ifIndex";
          }
          description
           "[Automatically generated leaf for a notification.]";
        }
      }

      container linkDown-ifAdminStatus {
        config false;
        leaf ifIndex {
          type leafref {
            path "/if-mib:interfaces/if-mib:ifTable" +
                 "/if-mib:ifEntry/if-mib:ifIndex";
          }
          description
           "[Automatically generated leaf for a notification.]";
        }
        leaf ifAdminStatus {
          type enumeration {
            enum up      { value 1; }
            enum down    { value 2; }
            enum testing { value 3; }
          }
          description
           "The desired state of the interface.  The testing(3) state
            indicates that no operational packets can be passed.  When a
            managed system initializes, all interfaces start with
            ifAdminStatus in the down(2) state.  As a result of either
            explicit management action or per configuration information
            retained by the managed system, ifAdminStatus is then
            changed to either the up(1) or testing(3) states (or remains
            in the down(2) state).";
        }
      }

      container linkDown-ifOperStatus {
        config false;
        leaf ifIndex {
          type leafref {
            path "/if-mib:interfaces/if-mib:ifTable" +
                 "/if-mib:ifEntry/if-mib:ifIndex";
```

```
          }
          description
           "[Automatically generated leaf for a notification.]";
        }
        leaf ifOperStatus {
          type enumeration {
            enum up             { value 1; }
            enum down           { value 2; }
            enum testing        { value 3; }
            enum unknown        { value 4; }
            enum dormant        { value 5; }
            enum notPresent     { value 6; }
            enum lowerLayerDown { value 7; }
          }
          description
           "The current operational state of the interface.  The
            testing(3) state indicates that no operational packets can
            be passed.  If ifAdminStatus is down(2) then ifOperStatus
            should be down(2).  If ifAdminStatus is changed to up(1)
            then ifOperStatus should change to up(1) if the interface is
            ready to transmit and receive network traffic; it should
            change to dormant(5) if the interface is waiting for
            external actions (such as a serial line waiting for an
            incoming connection); it should remain in the down(2) state
            if and only if there is a fault that prevents it from going
            to the up(1) state; it should remain in the notPresent(6)
            state if the interface has missing (typically, hardware)
            components.";
        }
      }
    }
```

11. YANG Language Extension Definition

This section defines some YANG extension statements that can be used to carry additional information from the original SMIv2 module into the YANG module. The YANG module references [RFC2578] and [RFC2579].

<CODE BEGINS> file "ietf-yang-smiv2@2011-03-14.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) 2011 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 XXXX; see
    the RFC itself for full legal notices.";
    // RFC Ed.: replace XXXX with actual RFC number and remove this note

  // RFC Ed.: please update the date to the date of publication
```

```
  revision 2011-03-14 {
    description
     "Initial revision.";
    reference
     "RFC XXXX: Translation of SMIv2 MIB Modules to YANG Modules";
    // RFC Ed.: replace XXXX with actual RFC number and remove this note
  }

  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
     "RFC2578: 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
     "RFC2579: 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";
    reference
     "RFC2578: 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.";
    reference
     "RFC2578: Structure of Management Information Version 2 (SMIv2)";
  }

}
```

<CODE ENDS>

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 XXXX
```

13. Security Considerations

This document defines a translation of the SMIv2 data modeling language to the YANG data modeling language. The translation itself has no security impact on the Internet.

Users of translated SMIv2 data models that have been published as RFCs should consult the security considerations of the respective RFCs. In addition, the security considerations for the NETCONF protocol [RFC4741] should be consulted to understand how NETCONF protects potentially sensitive information.

14. References

14.1. 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.

14.2. 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.

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

[RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

Appendix A. Changes from 01 to 02

- Preserving the SMIv2 nesting instead of a flat translation.
- Inlined the examples to avoid page flipping exercises.
- Clarifications and several editorial improvements.

Appendix B. Changes from 00 to 01

- Translation is config false; top-level container are marked as config false.
- Revised the overall document structure, added a YANG module for the definition of YANG extensions (smiv2:oid, smiv2:display-hint, smiv2:max-access, smiv2:defval), moved the IF-MIB example into an appendix.
- Alignment with RFC 6020 and RFC 6021.
- Started to use [[TODO]] markers inside the text instead of maintaining a TODO list as an appendix.

Author's Address

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de