

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: December 4, 2011

A. Bierman  
Brocade  
M. Bjorklund  
Tail-f Systems  
June 2, 2011

YANG Data Model for System Management  
draft-bierman-netmod-system-mgmt-00

Abstract

This document defines a YANG data model for the configuration and identification of the management system of a device.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 4, 2011.

Copyright Notice

Copyright (c) 2011 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.

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
1.1.1. Terms . . . . .	3
2. Objectives . . . . .	4
2.1. System Identification . . . . .	4
2.2. System Time Management . . . . .	4
2.3. User Authentication . . . . .	4
3. System Data Model . . . . .	5
3.1. User Authentication Model . . . . .	5
3.2. SSH Public Key Authentication . . . . .	5
3.3. Local User Password Authentication . . . . .	6
3.4. RADIUS Password Authentication . . . . .	6
4. System YANG module . . . . .	7
5. IANA Considerations . . . . .	20
6. Security Considerations . . . . .	21
7. Normative References . . . . .	22
Authors' Addresses . . . . .	23

## 1. Introduction

This document defines a YANG [RFC6020] data model for the configuration and identification of the management system of a device.

Devices which are managed by NETCONF and perhaps other mechanisms have common properties which need to be configured and monitored in a standard way.

The YANG module defined in this document provides the following features:

- o system administrative data configuration
- o system identification monitoring
- o system time-of-day configuration and monitoring
- o user authentication configuration
- o local users configuration

### 1.1. Terminology

The keywords "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].

#### 1.1.1. Terms

The following terms are used within this document:

- o system: This term refers to the embodiment of the entire set of management interfaces that a single NETCONF server is supporting at a given moment. The set of physical entities managed by a single NETCONF server can be static or it can change dynamically.

## 2. Objectives

### 2.1. System Identification

There are many common properties used to identify devices, operating systems, software versions, etc. that need to be supported in the system data module. These objects are defined as operational data and intended to be specific to the device vendor.

Some user-configurable administrative strings are also provided such as the system location and description.

### 2.2. System Time Management

The management of the date and time used by the system must be supported. Use of one or more NTP servers to automatically set the system date and time must be possible. Utilization of the Timezone database [I-D.lear-iana-timezone-database] must also be supported.

### 2.3. User Authentication

The authentication mechanism must support password authentication over RADIUS, to support deployment scenarios with centralized authentication servers. Additionally, local users must be supported, for scenarios when no centralized authentication server exists, or for situations where the centralized authentication server cannot be reached from the device.

Since the mandatory transport protocol for NETCONF is SSH [I-D.ietf-netconf-rfc4742bis] the authentication model must support SSH's "publickey" and "password" authentication methods [RFC4252].

The model for authentication configuration should be flexible enough to support authentication methods defined by other standard documents or by vendors.

### 3. System Data Model

[FIXME: this section currently just talks about authentication. Add description of the rest of the data model, like we do in snmp-cfg? Otherwise, rename this section to Authentication ...]

[FIXME: introduce a set of submodules to allow for future enhancements of the system data model?]

#### 3.1. User Authentication Model

This document defines three authentication methods for use with NETCONF:

- o publickey for local users over SSH
- o password for local users over any transport
- o password for RADIUS users over any transport

Additional methods may be defined by other standard documents or by vendors.

This document defines two optional YANG features, 'local-users' and 'radius', which the server advertises to indicate support for configuring local users on the device, and for configuring RADIUS access, respectively.

The authentication parameters defined in this document are primarily used to configure authentication of NETCONF users, but MAY also be used by other interfaces, e.g., a Command Line Interface or a Web-based User Interface.

#### 3.2. SSH Public Key Authentication

If the NETCONF server advertises the 'local-users' feature, configuration of local users and their SSH public keys is supported in the /system/authentication/user list.

Public key authentication is requested by the SSH client. The SSH server looks up the user name provided by the client in the /system/authentication/user list, and verifies the key as described in [RFC4253].

If the 'local-users' feature is supported, then when a NETCONF client starts an SSH session towards the server, using the "publickey" authentication 'method name' [RFC4252], the SSH server looks up the user name given in the SSH authentication request in the /system/

authentication/user list,

### 3.3. Local User Password Authentication

If the NETCONF server advertises the 'local-users' feature, configuration of local users and their passwords is supported in the /system/authentication/user list.

For NETCONF transport protocols that support password authentication, the leaf-list 'user-authentication-order' is used to control if local user password authentication should be used.

In SSH, password authentication is requested by the client. Other NETCONF transport protocols may also support password authentication.

When local user password authentication is requested, the NETCONF transport looks up the user name provided by the client in the /system/ authentication/user list, and verifies the password.

### 3.4. RADIUS Password Authentication

If the NETCONF server advertises the 'radius' feature, the device supports user authentication RADIUS.

For NETCONF transport protocols that support password authentication, the leaf-list 'user-authentication-order' is used to control if RADIUS password authentication should be used.

In SSH, password authentication is requested by the client. Other NETCONF transport protocols may also support password authentication.

#### 4. System YANG module

RFC Ed.: update the date below with the date of RFC publication and remove this note.

This YANG module imports YANG extensions from ... and references ...  
[Editor's Note: add proper references]

<CODE BEGINS> file "ietf-system@2011-06-02.yang"

```
module ietf-system {
  namespace "urn:ietf:params:xml:ns:yang:ietf-system";
  prefix "sys";

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-netconf-acm {
    prefix nacm;
  }

  import ietf-yang-types {
    prefix yang;
  }

  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:    Andy Bierman
               <mailto:andy.bierman@brocade.com>

    Editor:    Martin Bjorklund
               <mailto:mbj@tail-f.com>";

  description
    "This module contains a collection of YANG definitions for the
    configuration and identification of the management system of a
```

device.

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.: update the date below with the date of RFC publication
// and remove this note.
revision 2011-06-02 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for System Management";
}

/*
 * Typedefs
 */

typedef crypt-hash {
  type string {
    pattern "$0$.* | $1|5|6$[a-zA-Z0-9./]{2,16}$.*";
  }
  description
    "The crypt-hash type is used to store passwords using
    a hash function. This type is implemented in various UNIX
    systems as the function crypt(3).

    When a clear text value is set to a leaf of this type, the
    server calculates a password hash, and stores the result
    in the datastore. Thus, the password is never stored in
    clear text.

    When a leaf of this type is read, the stored password hash is
    returned.
```



A value of this type matches one of the forms:

```
$0$<clear text password>
$<id>$<salt>$<password hash>
```

The '\$0\$' prefix signals that the value is clear text. When such a value is received by the server, a hash value is calculated, and the string '\$<id>\$<salt>\$' is prepended to the result, where <salt> is a random 2-16 characters long salt used to generate the digest. This value is stored in the configuration data store.

If a value starting with '\$<id>\$<salt>\$' is received, the server knows that the value already represents a hashed value, and stores it as is in the data store.

When a server needs to verify a password given by a user, it finds the stored password hash string for that user, extracts the salt, and calculates the hash with the salt and given password as input. If the calculated hash value is the same as the stored value, the password given by the client is correct.

This type defines the following hash functions:

id	hash function	feature
1	MD5	crypt-hash-md5
5	SHA-256	crypt-hash-sha-256
6	SHA-512	crypt-hash-sha-512

The server indicates support for the different hash functions by advertising the corresponding feature."

reference

"Wikipedia: [http://en.wikipedia.org/wiki/Crypt\\_\(Unix\)](http://en.wikipedia.org/wiki/Crypt_(Unix))

RFC 1321: The MD5 Message-Digest Algorithm

FIPS.180-3.2008: Secure Hash Standard";

}

/\*

\* Features

\*/

feature authentication {

description

"Indicates that the device can be configured  
to do authentication of users.";

}

```
feature radius {
  if-feature authentication;
  description
    "Indicates that the device can be
    configured to act as a NAS and authenticate users
    with RADIUS.";
  reference
    "RFC 2865: Remote Authentication Dial In User Service (RADIUS)
    RFC 5607: Remote Authentication Dial-In User Service (RADIUS)
    Authorization for Network Access Server (NAS)
    Management";
}

feature local-users {
  if-feature authentication;
  description
    "Indicates that the device supports
    local user authentication.";
}

feature crypt-hash-md5 {
  description
    "Indicates that the device supports the MD5
    hash function in 'crypt-hash' values";
  reference "RFC 1321: The MD5 Message-Digest Algorithm";
}

feature crypt-hash-sha-256 {
  description
    "Indicates that the device supports the SHA-256
    hash function in 'crypt-hash' values";
  reference "FIPS.180-3.2008: Secure Hash Standard";
}

feature crypt-hash-sha-512 {
  description
    "Indicates that the device supports the SHA-512
    hash function in 'crypt-hash' values";
  reference "FIPS.180-3.2008: Secure Hash Standard";
}

feature ntp {
  description
    "Indicates that the device can be configured
    to use one or more NTP servers to set the
    system date and time.";
}
```

```
feature tz-database {
  description
    "Indicates that the local timezone on the device
     can be configured to use the TZ database
     to set the timezone and manage daylight savings time.";
  reference
    "TZ Database  http://www.twinsun.com/tz/tz-link.htm
     Maintaining the Timezone Database
     http://www.ietf.org/id/draft-lear-iana-timezone-database-04.txt
    ";
}

feature tz-enumeration {
  description
    "Indicates that the local timezone on the device
     can be configured using the timezone enumeration
     strings as an alias for an UTC offset.";
  reference
    "Wikipedia: http://en.wikipedia.org/wiki/"
    + "List_of_time_zone_abbreviations";
}

/*
 * Identities
 */

identity authentication-method {
  description
    "Base identity for user authentication methods.";
}

identity radius {
  base authentication-method;
  description
    "Indicates user authentication using RADIUS.";
  reference
    "RFC 2865: Remote Authentication Dial In User Service (RADIUS)
     RFC 5607: Remote Authentication Dial-In User Service (RADIUS)
     Authorization for Network Access Server (NAS)
     Management";
}

identity local-users {
  base authentication-method;
  description
    "Indicates password-based authentication of locally
     configured users.";
}
```

```
/*
 * Top-level container
 */

container system {
  description
    "System group configuration.";

  leaf contact {
    type string {
      length "0..255";
    }
    default "";
    reference
      "RFC 3418 - Management Information Base (MIB) for the
       Simple Network Management Protocol (SNMP)
       SNMPv2-MIB.sysContact";
  }

  leaf name {
    type string {
      length "0..255";
    }
    default "";
    reference
      "RFC 3418 - Management Information Base (MIB) for the
       Simple Network Management Protocol (SNMP)
       SNMPv2-MIB.sysName";
  }

  leaf location {
    type string {
      length "0..255";
    }
    default "";
    reference
      "RFC 3418 - Management Information Base (MIB) for the
       Simple Network Management Protocol (SNMP)
       SNMPv2-MIB.sysLocation";
  }

  container platform {
    description
      "Contains vendor-specific information for
       identifying the system platform and operating system.";
    reference
      "GNU coreutils homepage:
       http://www.gnu.org/software/coreutils

```

```
Wikipedia: http://en.wikipedia.org/wiki/Uname";

config false;

leaf os-name {
  type string;
  description
    "The name of the operating system in use,
    for example 'Linux'";
  reference
    "uname --kernel-name";
}

leaf os-release {
  type string;
  description
    "The current release level of the operating
    system in use. This string MAY indicate
    the OS source code revision.";
  reference
    "uname --kernel-release";
}

leaf os-version {
  type string;
  description
    "The current version level of the operating
    system in use. This string MAY indicate
    the specific OS build date and target variant
    information.";
  reference
    "uname --kernel-version";
}

leaf machine {
  type string;
  description
    "A vendor-specific identifier string representing
    the hardware in use.";
  reference
    "uname --machine";
}

leaf nodename {
  type string;
  description
    "The host name of this system.";
  reference
```

```
        "uname --nodename";
    }
}

container clock {
    description
        "Configuration and monitoring of the system
        date and time properties.";

    leaf current-datetime {
        description
            "The current system date and time.";
        type yang:date-and-time;
        config false;
    }

    leaf boot-datetime {
        description
            "The system date and time when the NETCONF
            server last restarted.";
        type yang:date-and-time;
        config false;
    }

    choice timezone-info {
        description
            "Configure the system timezone information.";

        leaf tz-database-id {
            if-feature tz-database;
            description
                "The TZ database location identifier string
                to use for the system, such as 'Europe/Stockholm'.";
            type string;
        }

        leaf tz-enumeration-id {
            if-feature tz-enumeration;
            description
                "The timezone enumeration string to use
                for the system, such as 'CET'.";
            type string;
            // FIXME: use TimezoneEnum typedef instead
            // see http://en.wikipedia.org/wiki/
            //      List_of_time_zone_abbreviations
        }

        leaf utc-offset {
```

```
        description
            "The number of minutes to add to UTC time to
            identify the timezone for this system.
            For example, 'UTC - 8:00 hours' would be
            represented as '-480'.";
        type int16 {
            range "-1439 .. 1439";
        }
    }
}

container ntp {
    if-feature ntp;

    description
        "Configuration of the NTP client.";

    leaf use-ntp {
        description
            "Indicates that the system should attempt
            to synchronize the system clock with an
            NTP server from the 'ntp-server' list.";
        type boolean;
        default true;
    }

    list ntp-server {
        description
            "List of NTP servers to use for
            system clock synchronization.  If 'use-ntp'
            is 'true', then the system will attempt to
            contact and utilize the specified NTP servers.";

        key address;

        leaf address {
            description
                "The IP address or domain name of the NTP server.";
            type inet:host;
        }

        // TBD: add more parameters here
        // and/or vendors can add parameters via augment
    }
}

container dns {
```

```
description
  "Configuration of the DNS resolver.

  The 'domain' keyword of /etc/resolv.conf is not supported,
  since it is equivalent to 'search' with a single domain.";

  leaf-list search {
    type inet:host;
    ordered-by user;
  }
  leaf-list server {
    type inet:ip-address;
    ordered-by user;
    description
      "Addresses of the name servers that the resolver should
      query.

      Implementations MAY limit the number of entries in this
      leaf list.";
  }
  container options {
    description
      "Resolver options. The set of available options has been
      limited to those that are generally available across
      different resolver implementations, and generally useful.";
    leaf ndots {
      type uint8;
      default "1";
    }
    leaf timeout {
      type uint8;
      units "seconds";
      default "5";
    }
    leaf attempts {
      type uint8;
      default "2";
    }
  }
}

container authentication {
  nacm:secure;
  if-feature authentication;

  description
    "The authentication configuration subtree.";
```



```
leaf-list user-authentication-order {
  type identityref {
    base authentication-method;
  }
  must '(. = "sys:radius" and ../radius/server) or'
    + '(. != "sys:radius")' {
    error-message
      "When 'radius' is used, a radius server
      must be configured.";
  }
  ordered-by user;

  description
    "When the device authenticates a user with
    a password, it tries the authentication methods in this
    leaf-list in order.  If authentication with one method
    fails, the next method is used.  If no method succeeds,
    the user is denied access.

    If the 'radius' feature is advertised by the NETCONF
    server, the 'radius' identity can be added to this
    list.

    If the 'local-users' feature is advertised by the
    NETCONF server, the 'local-users' identity can be
    added to this list.";
}

container radius {
  if-feature radius;

  description
    "The RADIUS configuration for authentication.";

  list server {
    key address;
    ordered-by user;

    description
      "The RADIUS server configuration used by
      the device.";

    leaf address {
      type inet:host;
      description
        "The address of the RADIUS server.";
    }
    leaf port {
```

```
        type inet:port-number;
        default "1812";
        description
            "The port number of the RADIUS server.";
    }
    leaf shared-secret {
        type string;
        nacm:very-secure;
        description
            "The shared secret which is known to both the RADIUS
            client and server.";
        reference
            "RFC 2865: Remote Authentication Dial In User Service";
    }
}

container options {
    description
        "RADIUS client options.";

    leaf timeout {
        type uint8;
        units "seconds";
        default "5";
        description
            "The number of seconds the device will wait for a
            response from a RADIUS server before trying with a
            different server.";
    }
    leaf attempts {
        type uint8;
        default "2";
        description
            "The number of times the device will send a query to
            the RADIUS servers before giving up.";
    }
}

list user {
    if-feature local-users;
    key name;

    description
        "The list of local users configured on this device.";

    leaf name {
        type string;
        description
```

```
        "The user name string identifying this entry.";
    }
    leaf password {
        type crypt-hash;
        description
            "The password for this entry.";
    }
    leaf ssh-dsa {
        type binary;
        description
            "The public DSA key for this entry.";
    }
    leaf ssh-rsa {
        type binary;
        description
            "The public RSA key for this entry.";
    }
}
}
}

rpc set-current-datetime {
    nacm:secure;
    description
        "Manually set the /system/clock/current-datetime leaf
        to the specified value.

        If the /system/ntp/ntp-in-use leaf exists and
        is set to 'true', then this operation will
        fail with error-tag 'operation-failed',
        and error-app-tag value of 'ntp-active'";
    input {
        leaf current-datetime {
            description
                "The current system date and time.";
            type yang:date-and-time;
            mandatory true;
        }
    }
}
}
```

<CODE ENDS>

## 5. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

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

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-system
namespace:	urn:ietf:params:xml:ns:yang:ietf-system
prefix:	sys
reference:	RFC XXXX

## 6. Security Considerations

TBD.

## 7. Normative References

- [I-D.ietf-netconf-rfc4742bis]  
Wasserman, M. and T. Goddard, "Using the NETCONF Configuration Protocol over Secure Shell (SSH)", draft-ietf-netconf-rfc4742bis-08 (work in progress), March 2011.
- [I-D.lear-iana-timezone-database]  
Lear, E. and P. Eggert, "IANA Procedures for Maintaining the Timezone Database", draft-lear-iana-timezone-database-04 (work in progress), May 2011.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC4252] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.
- [RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

Authors' Addresses

Andy Bierman  
Brocade

Email: andy.bierman@brocade.com

Martin Bjorklund  
Tail-f Systems

Email: mbj@tail-f.com





Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: November 21, 2011

M. Bjorklund  
Tail-f Systems  
May 20, 2011

A YANG Data Model for IP Configuration  
draft-bjorklund-netmod-ip-cfg-00

## Abstract

This document defines a YANG data model for configuration of IP addresses on network interfaces.

## Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 21, 2011.

## Copyright Notice

Copyright (c) 2011 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.

## Table of Contents

1. Introduction . . . . .	3
2. IP Data Model . . . . .	4
3. IP Address YANG Module . . . . .	5
4. IANA Considerations . . . . .	8
5. Security Considerations . . . . .	9
6. Normative References . . . . .	10
Appendix A. Example: NETCONF <get-config> reply . . . . .	11
Author's Address . . . . .	12

## 1. Introduction

This document defines a YANG [RFC6020] data model for configuration of IP addresses on network interfaces.

```
+-----+
| Open Question                                     |
+-----+
| What is the proper scope of this document? IP addresses only, or |
| IP configuration in general?                                   |
+-----+
```

The keywords "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. IP Data Model

The module "ietf-ip" augments the "interface" list defined in the "ietf-interfaces" module [I-D.ietf-netmod-interfaces-cfg] with the following nodes:

```

+--rw if:interfaces
  +--rw if:interface [name]
    ...
    +--rw ipv4
      | +--rw address [ip]
      |   +--rw ip                inet:ipv4-address
      |   +--rw prefix-length?    uint8
    +--rw ipv6
      | +--rw address [ip]
      |   +--rw ip                inet:ipv6-address
      |   +--rw prefix-length?    uint8
```

The data model defines two containers, "ipv4" and "ipv6", representing the IPv4 and IPv6 address families. In each container, there is a list of manually configured addresses.

### 3. IP Address YANG Module

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-ip@2011-05-20.yang"

```
module ietf-ip {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-ip";  
    prefix ip;  
  
    import ietf-interfaces {  
        prefix if;  
    }  
    import ietf-inet-types {  
        prefix inet;  
    }  
  
    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:    Martin Bjorklund  
                  <mailto:mbj@tail-f.com>";  
  
    description  
        "This module contains a collection of YANG definitions for  
        configuring IP addresses on network interfaces.  
  
        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.: update the date below with the date of RFC publication
// and remove this note.
revision 2011-05-20 {
    description
        "Initial revision.";
    reference
        "RFC XXXX: A YANG Data Model for IP Configuration";
}

augment "/if:interfaces/if:interface" {
    container ipv4 {
        description
            "Parameters for the IPv4 address familiy.";
        list address {
            key "ip";
            description
                "The list of manually configured IPv4 addresses
                on the interface.";

            leaf ip {
                type inet:ipv4-address;
            }
            leaf prefix-length {
                type uint8 {
                    range "0..32";
                }
            }
        }
    }
    container ipv6 {
        description
            "Parameters for the IPv6 address familiy.";
        list address {
            key "ip";
            description
                "The list of manually configured IPv6 addresses
                on the interface.";

            leaf ip {
                type inet:ipv6-address;
            }
            leaf prefix-length {

```

```
        type uint8 {
            range "0..128";
        }
    }
}

<CODE ENDS>
```

#### 4. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

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

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-ip
namespace:	urn:ietf:params:xml:ns:yang:ietf-ip
prefix:	ip
reference:	RFC XXXX



## 5. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [I-D.ietf-netconf-4741bis]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [I-D.ietf-netconf-rfc4742bis].

There are a number of data nodes defined in the YANG module which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

<list subtrees and data nodes and state why they are sensitive>

Some of the readable data nodes in the YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

<list subtrees and data nodes and state why they are sensitive>

## 6. Normative References

- [I-D.ietf-netconf-4741bis]  
Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", draft-ietf-netconf-4741bis-10 (work in progress), March 2011.
- [I-D.ietf-netconf-rfc4742bis]  
Wasserman, M. and T. Goddard, "Using the NETCONF Configuration Protocol over Secure Shell (SSH)", draft-ietf-netconf-rfc4742bis-08 (work in progress), March 2011.
- [I-D.ietf-netmod-interfaces-cfg]  
Bjorklund, M., "A YANG Data Model for Interface Configuration", draft-ietf-netmod-interfaces-cfg-01 (work in progress), May 2011.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

## Appendix A. Example: NETCONF &lt;get-config&gt; reply

This section gives an example of a reply to the NETCONF <get> request for a device that implements the example data models above.

```
<rpc-reply
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="101">
  <data>
    <interfaces
      xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type>ethernetCsmacd</type>
        <location>0</location>
        <if-index>2</if-index>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>192.0.2.1</ip>
            <prefix-length>24</prefix-length>
          </address>
        </ipv4>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

Author's Address

Martin Bjorklund  
Tail-f Systems

Email: [mbj@tail-f.com](mailto:mbj@tail-f.com)



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 9, 2012

M. Bjorklund  
Tail-f Systems  
J. Schoenwaelder  
Jacobs University  
July 8, 2011

A YANG Data Model for SNMP Configuration  
draft-bjorklund-netmod-snmp-cfg-01

Abstract

This document defines a collection of YANG definitions for configuring SNMP engines.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2012.

Copyright Notice

Copyright (c) 2011 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.

## Table of Contents

1. Introduction . . . . .	3
2. Overview . . . . .	4
3. Data Model . . . . .	5
3.1. General Considerations . . . . .	5
3.2. Common Definitions . . . . .	5
3.3. Engine Configuration . . . . .	5
3.4. Target Configuration . . . . .	6
3.5. Notification Configuration . . . . .	6
3.6. Proxy Configuration . . . . .	7
3.7. Community Configuration . . . . .	8
3.8. User-based Security Model Configuration . . . . .	9
3.9. View-based Access Control Model Configuration . . . . .	11
4. Definitions . . . . .	12
4.1. Module 'ietf-snmp' . . . . .	12
4.2. Submodule 'ietf-snmp-common' . . . . .	13
4.3. Submodule 'ietf-snmp-engine' . . . . .	17
4.4. Submodule 'ietf-snmp-target' . . . . .	20
4.5. Submodule 'ietf-snmp-notification' . . . . .	24
4.6. Submodule 'ietf-snmp-proxy' . . . . .	28
4.7. Submodule 'ietf-snmp-community' . . . . .	31
4.8. Submodule 'ietf-snmp-usm' . . . . .	35
4.9. Submodule 'ietf-snmp-vacm' . . . . .	40
5. IANA Considerations . . . . .	46
6. Security Considerations . . . . .	48
7. Acknowledgments . . . . .	49
8. References . . . . .	50
8.1. Normative References . . . . .	50
8.2. Informative References . . . . .	50
Appendix A. Example configurations . . . . .	52
A.1. Engine Configuration Example . . . . .	52
A.2. Community Configuration Example . . . . .	52
A.3. User-based Security Model Configuration Example . . . . .	53
A.4. Target and Notification Configuration Example . . . . .	54
A.5. Proxy Configuration Example . . . . .	56
A.6. View-based Access Control Model Configuration Example . . . . .	56
Authors' Addresses . . . . .	58

## 1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of SNMP engines. The configuration model is consistent with the MIB modules defined in [RFC3411], [RFC3412], [RFC3413], [RFC3414], [RFC3415], [RFC3418], and [RFC3584] but takes advantage of YANG's ability to define hierarchical configuration data models. The structure of the model has been derived from existing proprietary configuration models implemented as command line interfaces.

The keywords "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. Overview

In order to preserve the modularity of SNMP, the YANG configuration data model is organized in a set of YANG submodules, all sharing the same module namespace. This allows to add configuration support for additional SNMP features while keeping the number of namespaces that have to be dealt with down to a minimum.

### 3. Data Model

#### 3.1. General Considerations

Most YANG nodes are mapped 1-1 to the corresponding MIB object. The "reference" statement is used to indicate which corresponding MIB object the YANG node is mapped to. When there is not a simple 1-1 mapping, the "description" statement explains the mapping.

#### 3.2. Common Definitions

The submodule "ietf-snmp-common" defines a set of common typedefs, features, and the top-level container "snmp". All configuration parameters defined in the other submodules are organized under this top-level container.

This submodule defines two YANG features:

proxy: A server implements this feature if it can act as an SNMP Proxy.

notification-filter: A server implements this feature if it supports SNMP notification filtering.

#### 3.3. Engine Configuration

The submodule "ietf-snmp-engine", which defines configuration parameters that are specific to SNMP engines, has the following structure:

```
+--rw snmp
  +--rw engine
    +--rw enabled?      boolean
    +--rw listen
      | +--rw udp [ip port]
      |   +--rw ip      inet:ip-address
      |   +--rw port    inet:port-number
    +--rw version
      | +--rw v1?      empty
      | +--rw v2c?     empty
      | +--rw v3?      empty
    +--rw engine-id?   snmp:engine-id
```

The leaf "/snmp/engine/enabled" can be used to enable/disable an SNMP engine. The container "/snmp/engine/listen" provides configuration of the transport endpoints the engine is listening to. This container is expected to be augmented for other transports. The "/snmp/engine/version" container can be used to enable/disable the

different message processing models.

### 3.4. Target Configuration

The submodule "ietf-snmp-target", which defines configuration parameters that correspond to the objects in SNMP-TARGET-MIB, has the following structure:

```
+--rw snmp
  +--rw target [name]
    +--rw name          snmp:identifier
    +--rw (transport)
      +--:(udp)
        +--rw udp
          +--rw ip          inet:ip-address
          +--rw port?       inet:port-number
          +--rw prefix-length? uint8
        +--rw tag*         snmp:identifier
      +--rw timeout?      uint32
      +--rw retries?      uint8
      +--rw (params)?
```

An entry in the list "/snmp/target" corresponds to an "snmpTargetAddrEntry".

The "snmpTargetAddrTDomain" and "snmpTargetAddrTAddress" objects are mapped to transport-specific YANG nodes. Each transport is configured as a separate case in the choice "transport". Currently, SNMP over UDP is supported.

In order to provide a simpler configuration model with less cross-references, the "target" list also inlines the "snmpTargetParamsEntry" pointed to by "snmpTargetAddrParams". This is accomplished with a choice "params", which is augmented by security model specific submodules, currently "ietf-snmp-community" (Section 3.7) and "ietf-snmp-usm" (Section 3.8).

The YANG model does not define a separate list that maps directly to "snmpTargetParamsTable". Since "snmpProxyTable" also has a reference to this table, "snmpProxyTable" also has a choice "params" which is augmented by security model specific submodules (Section 3.6).

### 3.5. Notification Configuration

The submodule "ietf-snmp-notification", which defines configuration parameters that correspond to the objects in SNMP-NOTIFICATION-MIB, has the following structure:

```

+--rw snmp
  +--rw notify [name]
    |   +--rw name      snmp:identifier
    |   +--rw tag       leafref
    |   +--rw type?     enumeration
  +--rw notify-filter-profile [name]
    |   +--rw name      snmp:identifier
    |   +--rw include*  wildcard-object-identifier
    |   +--rw exclude*  wildcard-object-identifier
  +--rw enable-authen-traps?    boolean

```

It also augments the "target" list defined in the "ietf-snmp-target" submodule (Section 3.4) with one leaf:

```

+--rw snmp
  +--rw target [name]
    ...
    +--rw notify-filter-profile?  leafref

```

An entry in the list "/snmp/notify" corresponds to an "snmpNotifyEntry".

An entry in the list "/snmp/notify-filter-profile" corresponds to an "snmpNotifyFilterProfileEntry". In the MIB, there is a sparse relationship between "snmpTargetParamsTable" and "snmpNotifyFilterProfileTable". In the YANG model, this sparse relationship is represented with a leafref leaf "notify-filter-profile" in the "/snmp/target" list, which refers to an entry in the "/snmp/notify-filter-profile" list.

The "snmpNotifyFilterTable" is represented as a list "filter" within the "/snmp/notify-filter-profile" list.

### 3.6. Proxy Configuration

The submodule "ietf-snmp-proxy", which defines configuration parameters that correspond to the objects in SNMP-PROXY-MIB, has the following structure:

```

+--rw snmp
  +--rw proxy [name]
    +--rw name      snmp:identifier
    +--rw type      enumeration
    +--rw context-engine-id  snmp:engine-id
    +--rw context-name?      snmp:context-name
    +--rw params-in
    |   +--rw (params)
    +--rw single-target-out?  leafref

```

```

    +---rw multiple-target-out?    leafref

```

An entry in the list `"/snmp/proxy"` corresponds to an `"snmpProxyEntry"`.

Like the `"target"` list (Section 3.4), the `"proxy"` list inlines the `"snmpTargetParamsEntry"` pointed to by `"snmpProxyTargetParamsIn"`. This is accomplished with a choice `"params"`, which is augmented by security model specific submodules, currently `"ietf-snmp-community"` (Section 3.7) and `"ietf-snmp-usm"` (Section 3.8).

### 3.7. Community Configuration

The submodule `"ietf-snmp-community"`, which defines configuration parameters that correspond to the objects in `SNMP-COMMUNITY-MIB`, has the following structure:

```

+---rw snmp
  +---rw community [index]
    +---rw index          snmp:identifier
    +---rw (name)?
      | +---:(text-name)
      | | +---rw text-name?      string
      | +---:(binary-name)
      | | +---rw binary-name?    binary
    +---rw security-name   snmp:security-name
    +---rw engine-id?      snmp:engine-id
    +---rw context?        snmp:context-name
    +---rw target-tag?     leafref

```

It also augments the `"/snmp/target/params"` and `"/snmp/proxy/params-in/params"` choices with nodes for the Community-Based Security Model used by `SNMPv1` and `SNMPv2c`:

```

+--rw snmp
  +--rw target [name]
    |   ...
    |   +--rw (params)?
    |   |   +--:(v1)
    |   |   |   +--rw v1
    |   |   |   |   +--rw community    leafref
    |   |   +--:(v2c)
    |   |   |   +--rw v2c
    |   |   |   |   +--rw community    leafref
    |   +--rw mms?      union
  +--rw proxy
    +--rw params-in
      +--rw params
        +--:(v1)
        |   +--rw v1
        |   |   +--rw community    leafref
        +--:(v2c)
        |   +--rw v2c
        |   |   +--rw community    leafref

```

An entry in the list `"/snmp/community"` corresponds to an `"snmpCommunityEntry"`.

When a case `"v1"` or `"v2c"` is chosen, it implies a `snmpTargetParamsMpmModel 0` (SNMPv1) or `1` (SNMPv2), and a `snmpTargetParamsSecurityModel 1` (SNMPv1) or `2` (SNMPv2), respectively. Both cases implies a `snmpTargetParamsSecurityLevel noAuthNoPriv`.

### 3.8. User-based Security Model Configuration

The submodule `"ietf-snmp-usm"`, which defines configuration parameters that correspond to the objects in `SNMP-USER-BASED-SM-MIB`, has the following structure:

```

+--rw snmp
  +--rw usm
    +--rw local
      |   +--rw user [name]
      |   |   +-- {common user params}
    +--rw remote [engine-id]
      +--rw engine-id    snmp:engine-id
      +--rw user [name]
      |   +-- {common user params}

```

The `"{common user params}"` are:

```

+--rw name      snmp:identifier
+--rw auth?
|   +--rw (protocol)
|   |   +--:(md5)
|   |   |   +--rw md5
|   |   |   +-- {common key params}
|   |   +--:(sha)
|   |   |   +--rw sha
|   |   |   +-- {common key params}
+--rw priv?
|   +--rw (protocol)
|   |   +--:(des)
|   |   |   +--rw des
|   |   |   +-- {common key params}
|   |   +--:(aes)
|   |   |   +--rw aes
|   |   |   +-- {common key params}

```

The "{common key params}" are:

```

+--rw (key-type)?
+--:(password)
|   +--rw password?   string
+--:(key)
|   +--rw key?        string

```

It also augments the `"/snmp/target/params"` and `"/snmp/proxy/params-in/params"` choices with nodes for the SNMP User-based Security Model.

```

+--rw snmp
|   +--rw target [name]
|   |   ...
|   |   +--rw (params)?
|   |   |   +--:(usm)
|   |   |   |   +--rw usm
|   |   |   |   |   +--rw user-name      snmp:security-name
|   |   |   |   |   +--rw security-level security-level
|   +--rw proxy [name]
|   |   ...
|   |   +--rw params-in
|   |   |   +--rw (params)
|   |   |   |   +--:(usm)
|   |   |   |   |   +--rw usm
|   |   |   |   |   |   +--rw user-name      snmp:security-name
|   |   |   |   |   |   +--rw security-level security-level

```

In the MIB, there is a single table with local and remote users,

indexed by the engine id and user name. In the YANG model, there is one list of local users, and a nested list of remote users.

In the MIB, there are several objects related to changing the authentication and privacy keys. These objects are not present in the YANG model. Instead, there is a choice between a password or a localized key. If a password is given, it is used by the server to calculate a localized key, which is stored in the configuration. The clear-text password is never stored. This implies that if the engine id is changed, all users keys need to be changed as well.

### 3.9. View-based Access Control Model Configuration

The submodule "ietf-snmp-vacm", which defines configuration parameters that correspond to the objects in SNMP-VIEW-BASED-ACM-MIB, has the following structure:

```

+--rw snmp
  +--rw vacm
    +--rw group [name]
      +--rw name          group-name
      +--rw member [security-name]
        +--rw security-name  snmp:security-name
        +--rw security-model* snmp:security-model
      +--rw access [context security-model security-level]
        +--rw context          snmp:context-name
        +--rw context-match?    enumeration
        +--rw security-model    snmp:security-model-or-any
        +--rw security-level    snmp:security-level
        +--rw read-view?        leafref
        +--rw write-view?       leafref
        +--rw notify-view?      leafref
    +--rw view [name]
      +--rw name          view-name
      +--rw include*       snmp:wildcard-object-identifier
      +--rw exclude*       snmp:wildcard-object-identifier

```

The "vacmSecurityToGroupTable" and "vacmAccessTable" are mapped to a structure of nested lists in the YANG model. Groups are defined in the list "/snmp/vacm/group", and for each group, there is a sublist "member", which maps to "vacmSecurityToGroupTable", and a sublist "access", which maps to "vacmAccessTable".

MIB views are defined in the list "/snmp/vacm/view", and for each MIB view, there is a leaf-list of included subtree families and a leaf-list of excluded subtree families. This is more compact and thus more readable representation of the "vacmViewTreeFamilyTable".



## 4. Definitions

### 4.1. Module 'ietf-snmp'

<CODE BEGINS> file "ietf-snmp.yang"

```
module ietf-snmp {  
  
  namespace "urn:ietf:params:xml:ns:yang:ietf-snmp";  
  prefix "snmp";  
  
  include ietf-snmp-common {  
    revision-date 2011-07-08;  
  }  
  include ietf-snmp-engine {  
    revision-date 2011-07-08;  
  }  
  include ietf-snmp-target {  
    revision-date 2011-07-08;  
  }  
  include ietf-snmp-notification {  
    revision-date 2011-07-08;  
  }  
  include ietf-snmp-proxy {  
    revision-date 2011-07-08;  
  }  
  include ietf-snmp-community {  
    revision-date 2011-07-08;  
  }  
  include ietf-snmp-usm {  
    revision-date 2011-07-08;  
  }  
  include ietf-snmp-vacm {  
    revision-date 2011-07-08;  
  }  
  
  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: Martin Bjorklund  
<mailto:mbj@tail-f.com>

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

description

"This module contains a collection of YANG definitions for configuring SNMP engines.

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.: update the date below with the date of RFC publication  
// and remove this note.

```
revision 2011-07-08 {  
  description  
    "Initial revision."  
  reference  
    "RFC XXXX: A YANG Data Model for SNMP Configuration";  
}
```

```
}
```

<CODE ENDS>

4.2. Submodule 'ietf-snmp-common'

<CODE BEGINS> file "ietf-snmp-common.yang"

```
submodule ietf-snmp-common {  
  
  belongs-to ietf-snmp {  
    prefix snmp;  
  }  
}
```

## 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: Martin Bjorklund

<<mailto:mbj@tail-f.com>>

Editor: Juergen Schoenwaelder

<<mailto:j.schoenwaelder@jacobs-university.de>>;

## description

"This submodule contains a collection of common YANG definitions for configuring SNMP engines.

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.: update the date below with the date of RFC publication  
// and remove this note.

```
revision 2011-07-08 {  
  description  
    "Initial revision."  
  reference  
    "RFC XXXX: A YANG Data Model for SNMP Configuration";  
}
```

```
/* Collection of SNMP features */

feature proxy {
  description
    "A server implements this feature if it can act as an
    SNMP Proxy";
}

feature notification-filter {
  description
    "A server implements this feature if it supports SNMP
    notification filtering.";
}

// FIXME: should we have v1 and v2c as features?

/* Collection of SNMP specific data types */

typedef admin-string {
  type string {
    length "0..255";
  }
  description
    "Represents and SnmpAdminString as defined in RFC 3411.

    Note that the size of an SnmpAdminString is measured in
    octets, not characters.";
  reference "SNMP-FRAMEWORK-MIB.SnmpAdminString";
}

typedef identifier {
  type admin-string {
    length "1..32";
  }
  description
    "Identifiers are used to name items in the SNMP configuration
    data store.";
}

typedef context-name {
  type admin-string {
    length "0..32";
  }
  description
    "The context type represents an SNMP context name.";
}

typedef security-name {
```

```
    type admin-string {
      length "1..32";
    }
    description
      "The security-name type represents an SNMP security name.";
    reference
      "";
  }

  typedef security-model {
    type union {
      type enumeration {
        enum v1 { value 1; }
        enum v2c { value 2; }
        enum usm { value 3; }
        //enum tsm { value 4; }
      }
      // FIXME: remove this? isn't it better to revise this
      // enum as need arises?
      type int32 {
        range "1..2147483647";
      }
    }
    reference
      "RFC3411: An Architecture for Describing SNMP Management
        Frameworks";
  }

  typedef security-model-or-any {
    type union {
      type enumeration {
        enum any { value 0; }
      }
      type security-model;
    }
    reference
      "RFC3411: An Architecture for Describing SNMP Management
        Frameworks";
  }

  typedef security-level {
    type enumeration {
      enum no-auth-no-priv { value 1; }
      enum auth-no-priv { value 2; }
      enum auth-priv { value 3; }
    }
    reference
      "RFC3411: An Architecture for Describing SNMP Management
```

```

        Frameworks";
    }

    typedef engine-id {
        type string {
            pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2}){4,31}';
        }
        description
            "The Engine ID specified as a list of colon-specified hexa-
            decimal octets e.g. '4F:4C:41:71'.";
        reference
            "RFC3411: An Architecture for Describing SNMP Management
            Frameworks";
    }

    typedef wildcard-object-identifier {
        type string;
        description
            "The wildcard-object-identifier type represents an SNMP object
            identifier where subidentifiers can be given either as a label,
            in numeric form, or a wildcard, represented by a *.";
    }

    container snmp {
        description
            "Top-level container for SNMP related configuration and
            status objects.";
    }
}

```

<CODE ENDS>

#### 4.3. Submodule 'ietf-snmp-engine'

<CODE BEGINS> file "ietf-snmp-engine.yang"

```

submodule ietf-snmp-engine {

    belongs-to ietf-snmp {
        prefix snmp;
    }

    import ietf-yang-types {
        prefix yang;
    }
    import ietf-inet-types {
        prefix inet;
    }
}

```

```
}

include ietf-snmp-common;

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: Martin Bjorklund
          <mailto:mbj@tail-f.com>

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

description
  "This submodule contains a collection of YANG definitions
  for configuring SNMP engines.

  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.: update the date below with the date of RFC publication
// and remove this note.

revision 2011-07-08 {
  description
    "Initial revision.";
```

```
reference
  "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp {

  container engine {

    description
      "Configuration of the SNMP engine.";

    leaf enabled {
      type boolean;
      default "false";
      description
        "Enables the SNMP engine.";
    }

    container listen {
      description
        "Configuration of the transport endpoints on which the
        engine listens. Submodules providing configuration for
        additional transports are expected to augment this
        container.";

      list udp {
        key "ip port";
        description
          "A list of IPv4 and IPv6 addresses and ports to which the
          engine listens.";

        leaf ip {
          type inet:ip-address;
          description
            "The IPv4 or IPv6 address on which the engine listens.";
        }
        leaf port {
          type inet:port-number;
          description
            "The UDP port on which the engine listens.";
        }
      }
    }

    container version {
      description
        "SNMP version used by the engine";
      leaf vl {
```



```

        type empty;
    }
    leaf v2c {
        type empty;
    }
    leaf v3 {
        type empty;
        must "../engine-id" {
            error-message
                "when v3 is configured, an engine-id must be set";
        }
    }
}

leaf engine-id {
    type snmp:engine-id;
    description
        "The local SNMP engine's administratively-assigned unique
        identifier.

        If this leaf is not set, the device automatically
        calculates an engine id, as described in RFC 3411. A
        server MAY initialize this leaf with the automatically
        created value.";
    reference "SNMP-FRAMEWORK-MIB.snmpEngineID";
}
}
}
}

```

<CODE ENDS>

#### 4.4. Submodule 'ietf-snmp-target'

<CODE BEGINS> file "ietf-snmp-target.yang"

```

submodule ietf-snmp-target {

    belongs-to ietf-snmp {
        prefix snmp;
    }

    import ietf-inet-types {
        prefix inet;
    }

    include ietf-snmp-common;
}

```

## 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: Martin Bjorklund

<<mailto:mbj@tail-f.com>>

Editor: Juergen Schoenwaelder

<<mailto:j.schoenwaelder@jacobs-university.de>>;

## description

"This submodule contains a collection of YANG definitions for configuring SNMP targets.

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.

## reference

"RFC3413: Simple Network Management Protocol (SNMP) Applications";

// RFC Ed.: update the date below with the date of RFC publication  
// and remove this note.

## revision 2011-07-08 {

## description

"Initial revision.";

## reference

```

    "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp {

  list target {
    key name;
    description
      "List of targets.";
    reference "SNMP-TARGET-MIB.snmpTargetAddrTable";

    leaf name {
      type snmp:identifier;
      description
        "Identifies the target.";
      reference "SNMP-TARGET-MIB.snmpTargetAddrName";
    }
    choice transport {
      mandatory true;
      description
        "Transport address of the target.

        The snmpTargetAddrTDomain and snmpTargetAddrTAddress
        objects are mapped to transport-specific YANG nodes. Each
        transport is configured as a separate case in this
        choice. Submodules providing configuration for additional
        transports are expected to augment this choice.";
      reference "SNMP-TARGET-MIB.snmpTargetAddrTDomain
        SNMP-TARGET-MIB.snmpTargetAddrTAddress";
      case udp {
        reference "SNMPv2-TM.snmpUDPDomain
          TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv4
          TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv4z
          TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv6
          TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv6z";
        container udp {
          leaf ip {
            type inet:ip-address;
            mandatory true;
            reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress";
          }
          leaf port {
            type inet:port-number;
            default 162;
            description
              "UDP port number";
            reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress";
          }
        }
      }
    }
  }
}

```

```

    leaf prefix-length {
        type uint8;
        description
            "The value of this leaf must match the value of
            ../snmp:ip. If ../snmp:ip contains an ipv4 address,
            this leaf must be less than or equal to 32. If it
            contains an ipv6 address, it must be less than or
            equal to 128.

            Note that the prefix-length is currently only used by
            the Community-based Security Model to filter incoming
            messages. Furthermore, the prefix-length filtering
            is not covering all possible filters supported by the
            corresponding MIB object.";
        reference "SNMP-COMMUNITY-MIB.snmpTargetAddrTMask";
    }
}
}
leaf-list tag {
    type snmp:identifier;
    description
        "List of tag values used to select target address.";
    reference "SNMP-TARGET-MIB.snmpTargetAddrTagList";
}
leaf timeout {
    type uint32;
    units "0.01 seconds";
    default 1500;
    description
        "Needed only if this target can receive InformRequest-PDUs.";
    reference "SNMP-TARGET-MIB.snmpTargetAddrTimeout";
}
leaf retries {
    type uint8;
    default 3;
    description
        "Needed only if this target can receive InformRequest-PDUs.";
    reference "SNMP-TARGET-MIB.snmpTargetAddrRetryCount";
}
choice params {
    // FIMXE: mandatory or not?

    description
        "This choice is augmented with case nodes containing
        security model specific configuration parameters. Each
        such case represents one entry in the

```

```

        snmpTargetParamsTable.

        When the snmpTargetAddrParams object contains a reference
        to a non-existing snmpTargetParamsEntry, this choice does
        not contain any case, and vice versa.";
        reference "SNMP-TARGET-MIB.snmpTargetAddrParams
                  SNMP-TARGET-MIB.snmpTargetParamsTable";
    }
}
}
}

<CODE ENDS>

```

#### 4.5. Submodule 'ietf-snmp-notification'

```
<CODE BEGINS> file "ietf-snmp-notification.yang"
```

```

submodule ietf-snmp-notification {

  belongs-to ietf-snmp {
    prefix snmp;
  }

  include ietf-snmp-common;
  include ietf-snmp-target;

  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:   Martin Bjorklund
               <mailto:mbj@tail-f.com>

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

  description
    "This submodule contains a collection of YANG definitions

```

for configuring SNMP notifications.

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.
```

```
reference
  "RFC3413: Simple Network Management Protocol (SNMP) Applications";
```

```
// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
```

```
revision 2011-07-08 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for SNMP Configuration";
}
```

```
augment /snmp:snmp {
```

```
  list notify {
    key name;
    description
      "Targets that will receive notifications.
```

```
    Entries in this lists are mapped 1-1 to entries in
    snmpNotifyTable, except that if an entry in snmpNotifyTable
    has a snmpNotifyTag for which no snmpTargetAddrEntry exists,
    then the snmpNotifyTable entry is not mapped to an entry in
    this list.";
```

```
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyTable";
```

```
  leaf name {
    type snmp:identifier;
    description
      "An arbitrary name for the list entry.";
```

```
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyName";
  }
  leaf tag {
    type leafref {
      path "/snmp/target/tag";
    }
    mandatory true;
    description
      "Target tag, selects a set of notification targets.";
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyTag";
  }
  leaf type {
    type enumeration {
      enum trap { value 1; }
      enum inform { value 2; }
    }
    default trap;
    description
      "Defines the notification type to be generated.";
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyType";
  }
}

list notify-filter-profile {
  if-feature snmp:notification-filter;
  key name;

  description
    "Notification filter profiles.

    The leaf /snmp/target/notify-filter-profile is used
    to associate a filter profile with a target.

    If an entry in this list is referred to by one or more
    /snmp/target/notify-filter-profile, each such
    notify-filter-profile is represented by one
    snmpNotifyFilterProfileEntry.

    If an entry in this list is not referred to by any
    /snmp/target/notify-filter-profile, the entry is not mapped
    to snmpNotifyFilterProfileTable.";
  reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileTable
    SNMP-NOTIFICATION-MIB.snmpNotifyFilterTable";

  leaf name {
    type snmp:identifier;
    description
      "Name of the filter profile";
```

```
        reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileName";
    }

    leaf-list include {
        type wildcard-object-identifier;
        description
            "A family of subtrees included in this filter.";
        reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterSubtree
            SNMP-NOTIFICATION-MIB.snmpNotifyFilterMask
            SNMP-NOTIFICATION-MIB.snmpNotifyFilterType";
    }

    leaf-list exclude {
        type wildcard-object-identifier;
        description
            "A family of subtrees excluded from this filter.";
        reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterSubtree
            SNMP-NOTIFICATION-MIB.snmpNotifyFilterMask
            SNMP-NOTIFICATION-MIB.snmpNotifyFilterType";
    }
}

leaf enable-authen-traps {
    type boolean;
    description
        "Indicates whether the SNMP entity is permitted to
        generate authenticationFailure traps.";
    reference "SNMPv2-MIB.snmpEnableAuthenTraps";
}

augment /snmp:snmp/snmp:target {
    reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileTable";
    leaf notify-filter-profile {
        if-feature snmp:notification-filter;
        type leafref {
            path "/snmp/notify-filter-profile/name";
        }
        description
            "This leafref leaf is used to represent the sparse relationship
            between the /snmp/target list and the
            /snmp/notify-filter-profile list.";

        reference "SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileName";
    }
}
}
```



<CODE ENDS>

#### 4.6. Submodule 'ietf-snmp-proxy'

<CODE BEGINS> file "ietf-snmp-proxy.yang"

```
submodule ietf-snmp-proxy {  
  belongs-to ietf-snmp {  
    prefix snmp;  
  }  
  
  include ietf-snmp-common;  
  include ietf-snmp-target;  
  
  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: Martin Bjorklund  
            <mailto:mbj@tail-f.com>  
  
    Editor: Juergen Schoenwaelder  
            <mailto:j.schoenwaelder@jacobs-university.de>";  
  
  description  
    "This submodule contains a collection of YANG definitions  
    for configuring SNMP proxies.  
  
    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.

reference
  "RFC3413: Simple Network Management Protocol (SNMP) Applications";

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.

revision 2011-07-08 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp {
  if-feature snmp:proxy;

  list proxy {
    key name;

    description
      "List of proxy parameters.";
    reference "SNMP-PROXY-MIB.snmpProxyTable";

    leaf name {
      type snmp:identifier;
      description
        "Identifies the proxy parameter entry.";
      reference "SNMP-PROXY-MIB.snmpProxyName";
    }
    leaf type {
      type enumeration {
        enum read;
        enum write;
        enum trap;
        enum inform;
      }
      mandatory true;
      reference "SNMP-PROXY-MIB.snmpProxyType";
    }
    leaf context-engine-id {
      type snmp:engine-id;
      mandatory true;
      reference "SNMP-PROXY-MIB.snmpProxyContextEngineID";
    }
  }
}
```

```
    }
    leaf context-name {
      type snmp:context-name;
      reference "SNMP-PROXY-MIB.snmpProxyContextName";
    }
    container params-in {
      choice params {
        mandatory true;
        description
          "This choice is augmented with case nodes containing
          security model specific configuration parameters. Each
          such case represents one entry in the
          snmpTargetParamsTable.

          When the snmpProxyTargetParamsIn object contains a
          reference to a non-existing snmpTargetParamsEntry, this
          choice does not contain any case, and vice versa.";
      }
      reference "SNMP-PROXY-MIB.snmpProxyTargetParamsIn";
    }
    leaf single-target-out {
      when "../type = read or ../type = write";
      type leafref {
        path "/snmp:snmp/snmp:target/snmp:name";
      }
      description
        "When the snmpProxySingleTargetOut object contains
        a value which does not select an snmpTargetAddrEntry,
        this leaf does not exist.";
      reference "SNMP-PROXY-MIB.snmpProxySingleTargetOut";
    }
    leaf multiple-target-out {
      when "../type = trap or ../type = inform";
      type leafref {
        path "/snmp:snmp/snmp:target/snmp:tag";
      }
      description
        "When the snmpProxyMultipleTargetOut object contains
        a value which does not select any snmpTargetAddrEntries,
        this leaf does not exist.";
      reference "SNMP-PROXY-MIB.snmpProxyMultipleTargetOut";
    }
  }
}
```

<CODE ENDS>

#### 4.7. Submodule 'ietf-snmp-community'

<CODE BEGINS> file "ietf-snmp-community.yang"

```
submodule ietf-snmp-community {  
  belongs-to ietf-snmp {  
    prefix snmp;  
  }  
  
  include ietf-snmp-common;  
  include ietf-snmp-target;  
  include ietf-snmp-proxy;  
  
  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: Martin Bjorklund  
            <mailto:mbj@tail-f.com>  
  
    Editor: Juergen Schoenwaelder  
            <mailto:j.schoenwaelder@jacobs-university.de>";  
  
  description  
    "This submodule contains a collection of YANG definitions  
    for configuring community-based SNMP."
```

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.

reference
  "RFC3584: Coexistence between Version 1, Version 2, and Version 3
    of the Internet-standard Network Management Framework";

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.

revision 2011-07-08 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

augment /snmp:snmp {

  list community {
    key index;

    description
      "List of communities";
    reference "SNMP-COMMUNITY-MIB.snmpCommunityTable";

    leaf index {
      type snmp:identifier;
      description
        "Index into the community list.";
      reference "SNMP-COMMUNITY-MIB.snmpCommunityIndex";
    }
    choice name {
      description
        "The community name, either specified as a string
        or as a binary. The binary name is used when the
        community name contains characters that are not legal
        in a string.

        If not set, the value of 'security-name' is operationally
        used as the snmpCommunityName.";
      reference "SNMP-COMMUNITY-MIB.snmpCommunityName";
      leaf text-name {
        type string;
        description

```

```
        "A community name that can be represented as a
        YANG string.";
    }
    leaf binary-name {
        type binary;
        description
            "A community name represented as a binary value.";
    }
}
leaf security-name {
    type snmp:security-name;
    mandatory true;
    description
        "The snmpCommunitySecurityName of this entry.";
    reference "SNMP-COMMUNITY-MIB.snmpCommunitySecurityName";
}
leaf engine-id {
    if-feature snmp:proxy;
    type snmp:engine-id;
    description
        "If not set, the value of the local SNMP engine is
        operationally used by the device.";
    reference "SNMP-COMMUNITY-MIB.snmpCommunityContextEngineID";
}
leaf context {
    type snmp:context-name;
    default "";
    description
        "The context in which management information is accessed
        when using the community string specified by this entry.";
    reference "SNMP-COMMUNITY-MIB.snmpCommunityContextName";
}
leaf target-tag {
    type leafref {
        path "/snmp/target/tag";
    }
    description
        "Used to limit access for this community to the specified
        targets.";
    reference "SNMP-COMMUNITY-MIB.snmpCommunityTransportTag";
}
}

grouping vl-target-params {
    container vl {
        description
            "SNMPv1 parameters type."
    }
}
```

```
    Represents snmpTargetParamsMPModel '0',
    snmpTargetParamsSecurityModel '1', and
    snmpTargetParamsSecurityLevel 'noAuthNoPriv'.
  leaf community {
    type leafref {
      path "/snmp/community/security-name";
    }
    mandatory true;
    reference "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
  }
}

grouping v2c-target-params {
  container v2c {
    description
      "SNMPv2 community parameters type.
      Represents snmpTargetParamsMPModel '1',
      snmpTargetParamsSecurityModel '2', and
      snmpTargetParamsSecurityLevel 'noAuthNoPriv'.";
    leaf community {
      type leafref {
        path "/snmp/community/security-name";
      }
      mandatory true;
      reference "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
    }
  }
}

augment /snmp:snmp/snmp:target/snmp:params {
  case v1 {
    uses v1-target-params;
  }
  case v2c {
    uses v2c-target-params;
  }
}

augment /snmp:snmp/snmp:proxy/snmp:params-in/snmp:params {
  case v1 {
    uses v1-target-params;
  }
  case v2c {
    uses v2c-target-params;
  }
}
```

```
augment /snmp:snmp/snmp:target {
  leaf mms {
    when "snmp:params/snmp:v1 or snmp:params/snmp:v2c";
    type union {
      type enumeration {
        enum "unknown";
      }
      type int32 {
        range "484..max";
      }
    }
    default "484";
    reference
      "SNMP-COMMUNITY-MIB.snmpTargetAddrMMS";
  }
}
```

<CODE ENDS>

#### 4.8. Submodule 'ietf-snmp-usm'

<CODE BEGINS> file "ietf-snmp-usm.yang"

```
submodule ietf-snmp-usm {

  belongs-to ietf-snmp {
    prefix snmp;
  }

  include ietf-snmp-common;
  include ietf-snmp-target;
  include ietf-snmp-proxy;

  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: Martin Bjorklund  
<mailto:mbj@tail-f.com>

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

description

"This submodule contains a collection of YANG definitions for configuring the User-based Security Model (USM) of SNMP.

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.

reference

"RFC3414: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)."

// RFC Ed.: update the date below with the date of RFC publication  
// and remove this note.

revision 2011-07-08 {

description

"Initial revision."

reference

"RFC XXXX: A YANG Data Model for SNMP Configuration";

}

grouping key {

choice key-type {

leaf password {

type string;

description

"If this leaf is set, the server uses its value to create a localized key, according to the algorithm described in RFC 3414. The resulting localized key is stored in the configuration, in the 'key' leaf. The clear-text password

is never stored, and thus never returned in a read operation.

Note that if the engine id is changed, the passwords for the engine's users need to be set again, in order to re-calculate the localized keys."

```

    }
    leaf key {
      type string {
        pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2})*';
      }
      description
        "Localized key specified as a list of colon-specified
        hexa-decimal octets";
    }
  }
}

grouping user-list {
  list user {
    key "name";

    reference "SNMP-USER-BASED-SM-MIB.usmUserTable";

    leaf name {
      type snmp:identifier {
        length "1..32";
      }
      reference "SNMP-USER-BASED-SM-MIB.usmUserName";
    }
    container auth {
      presence "enables authentication";
      description
        "Enables authentication of the user";
      choice protocol {
        mandatory true;
        reference "SNMP-USER-BASED-SM-MIB.usmUserAuthProtocol";
        container md5 {
          uses key;
          reference "SNMP-USER-BASED-SM-MIB.usmHMACMD5AuthProtocol";
        }
        container sha {
          uses key;
          reference "SNMP-USER-BASED-SM-MIB.usmHMACSHAAuthProtocol";
        }
      }
    }
  }
  container priv {

```

```
    must "../auth" {
      error-message
        "when privacy is used, authentication must also be used";
    }
    presence "enables encryption";
    description
      "Enables encryption of SNMP messages.";

    choice protocol {
      mandatory true;
      reference "SNMP-USER-BASED-SM-MIB.usmUserPrivProtocol";
      container des {
        uses key;
        reference "SNMP-USER-BASED-SM-MIB.usmDESPrivProtocol";
      }
      container aes {
        uses key;
        reference "SNMP-USM-AES-MIB.usmAesCfb128Protocol";
      }
    }
  }
}

augment /snmp:snmp {

  container usm {
    description
      "Configuration of the User-based Security Model";
    container local {
      uses user-list;
    }

    list remote {
      key "engine-id";

      leaf engine-id {
        type snmp:engine-id;
        reference "SNMP-USER-BASED-SM-MIB.usmUserEngineID";
      }

      uses user-list;
    }
  }
}

grouping usm-target-params {
  container usm {
```

```

    description
      "User based SNMPv3 parameters type.
      Represents snmpTargetParamsMModel '3'.";
    leaf user-name {
      type snmp:security-name;
      mandatory true;
      reference
        "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
    }
    leaf security-level {
      type security-level;
      mandatory true;
      reference
        "SNMP-TARGET-MIB.snmpTargetParamsSecurityLevel";
    }
  }
}

augment /snmp:snmp/snmp:target/snmp:params {
  case usm {
    uses usm-target-params;
  }
}

augment /snmp:snmp/snmp:proxy/snmp:params-in/snmp:params {
  case usm {
    uses usm-target-params;
  }
}

augment /snmp:snmp/snmp:target {
  // FIXME: choice between leaf discovery { type empty; }
  // and engine-id?
  leaf engine-id {
    type leafref {
      path "/snmp/usm/remote/engine-id";
    }
  }
  must '../usm/user-name' {
    error-message
      "When engine-id is set, usm/user-name must also be set.";
  }
  must '/snmp/usm/remote[engine-id=current()]/'
    + 'user[name=current()../usm/user-name]' {
    error-message
      "When engine-id is set, the usm/user-name must exist in
      the /snmp/usm/remote list for this engine-id.";
  }
  description

```

```
"Needed only if this target can receive InformRequest-PDUs
over SNMPv3.
```

```
    This object is not present in the SNMP MIBs. In
    RFC 3412, it is a implementation specific matter how this
    engine-id is handled.";
    reference "RFC 3412 7.1.9a";
  }
}
}

<CODE ENDS>
```

#### 4.9. Submodule 'ietf-snmp-vacm'

```
<CODE BEGINS> file "ietf-snmp-vacm.yang"
```

```
submodule ietf-snmp-vacm {

  belongs-to ietf-snmp {
    prefix snmp;
  }

  include ietf-snmp-common;

  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: Martin Bjorklund
            <mailto:mbj@tail-f.com>

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

  description
    "This submodule contains a collection of YANG definitions
    for configuring the View-based Access Control Model (VACM)
```

of SNMP.

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.
```

reference

"RFC3415: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)";

```
// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
```

```
revision 2011-07-08 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for SNMP Configuration";
}
```

```
typedef view-name {
  type snmp:identifier;
  description
    "The view-name type represents an SNMP VACM view name.";
}
```

```
typedef group-name {
  type snmp:identifier;
  description
    "The group-name type represents an SNMP VACM group name.";
}
```

```
augment /snmp:snmp {

  container vacm {
    description
      "Configuration of the View-based Access Control Model";
```

```
list group {
  key name;
  description
    "VACM Groups.

    This data model has a different structure than the MIB.
    Groups are explicitly defined in this list, and group
    members are defined in the 'member' list (mapped to
    vacmSecurityToGroupTable), and access for the group is
    defined in the 'access' list (mapped to
    vacmAccessTable).";
  reference "SNMP-VIEW-BASED-ACM-MIB.vacmSecurityToGroupTable
    SNMP-VIEW-BASED-ACM-MIB.vacmAccessTable";

  leaf name {
    type group-name;
    description
      "The name of this VACM group.";
    reference "SNMP-VIEW-BASED-ACM-MIB.vacmGroupName";
  }

  list member {
    key "security-name";
    min-elements 1;
    description
      "A member of this VACM group. According to VACM, every
      group must have at least one member.

      A certain combination of security-name and security-model
      MUST NOT be present in more than one group.";
    reference "SNMP-VIEW-BASED-ACM-MIB.vacmSecurityToGroupTable";

    leaf security-name {
      type snmp:security-name;
      description
        "The securityName of a group member.";
      reference "SNMP-VIEW-BASED-ACM-MIB.vacmSecurityName";
    }

    leaf-list security-model {
      type snmp:security-model;
      min-elements 1;
      description
        "The security models under which this security-name
        is a member of this group.";
      reference "SNMP-VIEW-BASED-ACM-MIB.vacmSecurityModel";
    }
  }
}
```

```
list access {
  key "context security-model security-level";
  description
    "Definition of access right for groups";
  reference "SNMP-VIEW-BASED-ACM-MIB.vacmAccessTable";

  leaf context {
    type snmp:context-name;
    description
      "The context (prefix) under which the access rights
      apply.";
    reference
      "SNMP-VIEW-BASED-ACM-MIB.vacmAccessContextPrefix";
  }

  leaf context-match {
    type enumeration {
      enum exact;
      enum prefix;
    }
    default exact;
    reference
      "SNMP-VIEW-BASED-ACM-MIB.vacmAccessContextMatch";
  }

  leaf security-model {
    type snmp:security-model-or-any;
    description
      "The security model under which the access rights
      apply.";
    reference
      "SNMP-VIEW-BASED-ACM-MIB.vacmAccessSecurityModel";
  }

  leaf security-level {
    type snmp:security-level;
    description
      "The minimum security level under which the access
      rights apply.";
    reference
      "SNMP-VIEW-BASED-ACM-MIB.vacmAccessSecurityLevel";
  }

  leaf read-view {
    type leafref {
      path "/snmp/vacm/view/name";
    }
    description
```



```
        "The name of the MIB view of the SNMP context
        authorizing read access. If this leaf does not
        exist in a configuration, it maps to a zero-length
        vacmAccessReadViewName.";
    reference
        "SNMP-VIEW-BASED-ACM-MIB.vacmAccessReadViewName";
}

leaf write-view {
    type leafref {
        path "/snmp/vacm/view/name";
    }
    description
        "The name of the MIB view of the SNMP context
        authorizing write access. If this leaf does not
        exist in a configuration, it maps to a zero-length
        vacmAccessWriteViewName.";
    reference
        "SNMP-VIEW-BASED-ACM-MIB.vacmAccessWriteViewName";
}

leaf notify-view {
    type leafref {
        path "/snmp/vacm/view/name";
    }
    description
        "The name of the MIB view of the SNMP context
        authorizing notify access. If this leaf does not
        exist in a configuration, it maps to a zero-length
        vacmAccessNotifyViewName.";
    reference
        "SNMP-VIEW-BASED-ACM-MIB.vacmAccessNotifyViewName";
}
}

list view {
    key name;
    description
        "Definition of MIB views.";
    reference
        "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyTable";

    leaf name {
        type view-name;
        description
            "The name of this VACM MIB view.";
        reference
```



## 5. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

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

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-snmp
namespace:	urn:ietf:params:xml:ns:yang:ietf-snmp
prefix:	snmp
reference:	RFC XXXX

The document registers the following YANG submodules in the YANG Module Names registry [RFC6020].

```
name:      ietf-snmp-common
parent:    ietf-snmp
reference:  RFC XXXX

name:      ietf-snmp-engine
parent:    ietf-snmp
reference:  RFC XXXX

name:      ietf-snmp-community
parent:    ietf-snmp
reference:  RFC XXXX

name:      ietf-snmp-notification
parent:    ietf-snmp
reference:  RFC XXXX

name:      ietf-snmp-target
parent:    ietf-snmp
reference:  RFC XXXX

name:      ietf-snmp-usm
parent:    ietf-snmp
reference:  RFC XXXX

name:      ietf-snmp-vacm
parent:    ietf-snmp
reference:  RFC XXXX
```

## 6. Security Considerations

The YANG module and submodules defined in this memo are designed to be accessed via the NETCONF protocol [I-D.ietf-netconf-4741bis]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [I-D.ietf-netconf-rfc4742bis].

There are a number of data nodes defined in the YANG module and submodules which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

<list subtrees and data nodes and state why they are sensitive>

Some of the readable data nodes in the YANG module and submodules may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

<list subtrees and data nodes and state why they are sensitive>

## 7. Acknowledgments

The authors want to thank David Spakes for his review and valuable comments.

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

### 8.2. Informative References

- [I-D.ietf-netconf-4741bis]  
Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", draft-ietf-netconf-4741bis-10 (work in progress), March 2011.
- [I-D.ietf-netconf-rfc4742bis]  
Wasserman, M. and T. Goddard, "Using the NETCONF Configuration Protocol over Secure Shell (SSH)", draft-ietf-netconf-rfc4742bis-08 (work in progress), March 2011.
- [RFC3411] Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, December 2002.
- [RFC3412] Case, J., Harrington, D., Presuhn, R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3412, December 2002.
- [RFC3413] Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, December 2002.
- [RFC3414] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC 3414, December 2002.
- [RFC3415] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3415, December 2002.

- [RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 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", BCP 74, RFC 3584, August 2003.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.



## Appendix A. Example configurations

## A.1. Engine Configuration Example

Below is an XML instance document showing a configuration of an SNMP engine listening on UDP port 161 on IPv4 and IPv6 endpoints and accepting SNMPv2c and SNMPv3 messages.

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <engine>
    <enabled>true</enabled>
    <listen>
      <udp>
        <ip>0.0.0.0</ip>
        <port>161</port>
      </udp>
      <udp>
        <ip>:::</ip>
        <port>161</port>
      </udp>
    </listen>
    <version>
      <v2c/>
      <v3/>
    </version>
    <engine-id>80:00:02:b8:04:61:62:63</engine-id>
  </engine>
</snmp>
```

## A.2. Community Configuration Example

Below is an XML instance document showing a configuration that maps the community name "public" to the security-name "community-public" on the local engine with the default context name. The target tag "community-public-access" filters the access to this community name.

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <community>
    <index>1</index>
    <text-name>public</text-name>
    <security-name>community-public</security-name>
    <target-tag>community-public-access</target-tag>
  </community>
</snmp>
```

[TODO: Add example for the target params table.]

## A.3. User-based Security Model Configuration Example

Below is an XML instance document showing the configuration of a local user "joey" who has no authentication or privacy keys. For the remote SNMP engine identified by the snmpEngineID '800002b804616263'H, two users are configured. The user "matt" has a localized SHA authentication key and the user "russ" has a localized SHA authentication key and an AES encryption key.

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <usm>
    <local>
      <user>
        <name>joey</name>
      </user>
    </local>
    <remote>
      <engine-id>00:00:00:00:00:00:00:00:00:00:02</engine-id>
      <user>
        <name>matt</name>
        <auth>
          <sha>
            <!--
              The 'key' value is split into two lines to match
              the RFC formatting rules.
            -->
            <key>66:95:fe:bc:92:88:e3:62:82:23:
              5f:c7:15:1f:12:84:97:b3:8f:3f</key>
          </sha>
        </auth>
      </user>
      <user>
        <name>russ</name>
        <auth>
          <sha>
            <!--
              The 'key' value is split into two lines to match
              the RFC formatting rules.
            -->
            <key>66:95:fe:bc:92:88:e3:62:82:23:
              5f:c7:15:1f:12:84:97:b3:8f:3f</key>
          </sha>
        </auth>
        <priv>
          <aes>
            <key>66:95:fe:bc:92:88:e3:62:82:23:5f:c7:15:1f:12:84</key>
          </aes>
        </priv>
      </user>
    </remote>
  </usm>
</snmp>
```

```
        </user>
      </remote>
    </usm>
    <target>
      <name>bluebox</name>
      <udp>
        <ip>2001:db8::abcd</ip>
        <port>161</port>
      </udp>
      <tag>blue</tag>
      <usm>
        <user-name>matt</user-name>
        <security-level>auth-no-priv</security-level>
      </usm>
    </target>
  </snmp>
```

#### A.4. Target and Notification Configuration Example

Below is an XML instance document showing the configuration of a notification generator application (see Appendix A of [RFC3413]). Note that the USM specific objects are defined in the `ietf-snmp-usm.yang` submodule.

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <target>
    <name>addr1</name>
    <udp>
      <ip>128.1.2.3</ip>
      <port>162</port>
    </udp>
    <tag>group1</tag>
    <usm>
      <user-name>joe</user-name>
      <security-level>auth-no-priv</security-level>
    </usm>
  </target>
  <target>
    <name>addr2</name>
    <udp>
      <ip>128.2.4.6</ip>
      <port>162</port>
    </udp>
    <tag>group1</tag>
    <usm>
      <user-name>joe</user-name>
      <security-level>auth-no-priv</security-level>
    </usm>
  </target>
  <target>
    <name>addr3</name>
    <udp>
      <ip>128.1.5.9</ip>
      <port>162</port>
    </udp>
    <tag>group2</tag>
    <usm>
      <user-name>bob</user-name>
      <security-level>auth-priv</security-level>
    </usm>
  </target>
  <notify>
    <name>group1</name>
    <tag>group1</tag>
    <type>trap</type>
  </notify>
  <notify>
    <name>group2</name>
    <tag>group2</tag>
    <type>trap</type>
  </notify>
</snmp>
```

## A.5. Proxy Configuration Example

[TODO]

## A.6. View-based Access Control Model Configuration Example

Below is an XML instance document showing the minimum-secure VACM configuration (see Appendix A of [RFC3415]).

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <vacm>
    <group>
      <name>initial</name>
      <member>
        <security-name>initial</security-name>
        <security-model>usm</security-model>
      </member>
      <access>
        <context></context>
        <security-model>usm</security-model>
        <security-level>no-auth-no-priv</security-level>
        <read-view>restricted</read-view>
        <notify-view>restricted</notify-view>
      </access>
      <access>
        <context></context>
        <security-model>usm</security-model>
        <security-level>auth-no-priv</security-level>
        <read-view>internet</read-view>
        <write-view>internet</write-view>
        <notify-view>internet</notify-view>
      </access>
    </group>
    <view>
      <name>initial</name>
      <include>1.3.6.1</include>
    </view>
    <view>
      <name>restricted</name>
      <include>1.3.6.1</include>
    </view>
  </vacm>
</snmp>
```

The following XML instance document shows the semi-secure VACM configuration (only the view configuration is different).

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <vacm>
    <group>
      <name>initial</name>
      <member>
        <security-name>initial</security-name>
        <security-model>usm</security-model>
      </member>
      <access>
        <context></context>
        <security-model>usm</security-model>
        <security-level>no-auth-no-priv</security-level>
        <read-view>restricted</read-view>
        <notify-view>restricted</notify-view>
      </access>
      <access>
        <context></context>
        <security-model>usm</security-model>
        <security-level>auth-no-priv</security-level>
        <read-view>internet</read-view>
        <write-view>internet</write-view>
        <notify-view>internet</notify-view>
      </access>
    </group>
    <view>
      <name>initial</name>
      <include>1.3.6.1</include>
    </view>
    <view>
      <name>restricted</name>
      <include>1.3.6.1</include>
    </view>
  </vacm>
</snmp>
```

Authors' Addresses

Martin Bjorklund  
Tail-f Systems

Email: [mbj@tail-f.com](mailto:mbj@tail-f.com)

Juergen Schoenwaelder  
Jacobs University

Email: [j.schoenwaelder@jacobs-university.de](mailto:j.schoenwaelder@jacobs-university.de)





Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: March 27, 2011

P. Shafer  
Juniper Networks  
September 23, 2010

An Architecture for Network Management using NETCONF and YANG  
draft-ietf-netmod-arch-10

## Abstract

The Network Configuration Protocol (NETCONF) gives access to native capabilities of the devices within a network, defining methods for manipulating configuration databases, retrieving operational data, and invoking specific operations. YANG provides the means to define the content carried via NETCONF, both data and operations. Using both technologies, standard modules can be defined to give interoperability and commonality to devices, while still allowing devices to express their unique capabilities.

This document describes how NETCONF and YANG help build network management applications that meet the needs of network operators.

## Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 27, 2011.

## Copyright Notice

Copyright (c) 2010 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. Origins of NETCONF and YANG . . . . .	4
2. Elements of the Architecture . . . . .	6
2.1. NETCONF . . . . .	6
2.1.1. NETCONF Transport Mappings . . . . .	8
2.2. YANG . . . . .	9
2.2.1. Constraints . . . . .	11
2.2.2. Flexibility . . . . .	12
2.2.3. Extensibility Model . . . . .	12
2.3. YANG Translations . . . . .	13
2.3.1. YIN . . . . .	14
2.3.2. DSDL (RELAX NG) . . . . .	14
2.4. YANG Types . . . . .	15
2.5. IETF Guidelines . . . . .	15
3. Working with YANG . . . . .	16
3.1. Building NETCONF- and YANG-based Solutions . . . . .	16
3.2. Addressing Operator Requirements . . . . .	17
3.3. Roles in Building Solutions . . . . .	20
3.3.1. Modeler . . . . .	20
3.3.2. Reviewer . . . . .	20
3.3.3. Device Developer . . . . .	20
3.3.4. Application Developer . . . . .	21
4. Modeling Considerations . . . . .	24
4.1. Default Values . . . . .	24
4.2. Compliance . . . . .	25
4.3. Data Distinctions . . . . .	26
4.3.1. Background . . . . .	26
4.3.2. Definitions . . . . .	27
4.3.3. Implications . . . . .	28
4.4. Direction . . . . .	29
5. Security Considerations . . . . .	30
6. IANA Considerations . . . . .	31
7. Normative References . . . . .	32
Author's Address . . . . .	34

## 1. Origins of NETCONF and YANG

Networks are increasing in complexity and capacity, as well as the density of the services deployed upon them. Uptime, reliability, and predictable latency requirements drive the need for automation. The problems with network management are not simple. They are complex and intricate. But these problems must be solved for networks to meet the stability needs of existing services while incorporating new services in a world where the growth of networks is exhausting the supply of qualified networking engineers.

In June of 2002, Internet Architecture Board (IAB) held a workshop on Network Management ([RFC3535]). The members of this workshop made a number of observations and recommendations for the IETF's consideration concerning the issues operators were facing in their network management-related work as well as issues they were having with the direction of the IETF activities in this area.

The output of this workshop was focused on current problems. The observations were reasonable and straight forward, including the need for transactions, rollback, low implementation costs, and the ability to save and restore the device's configuration data. Many of the observations give insight into the problems operators were having with existing network management solutions, such as the lack of full coverage of device capabilities and the ability to distinguish between configuration data and other types of data.

Based on these directions, the NETCONF working group was formed and the Network Configuration (NETCONF) protocol was created. This protocol defines a simple mechanism where network management applications, acting as clients, can invoke operations on the devices, which act as servers. The NETCONF specification ([RFC4741]) defines a small set of operations, but goes out of its way to avoid making any requirements on the data carried in those operations, preferring to allow the protocol to carry any data. This "data model agnostic" approach allows data models to be defined independently.

But lacking a means of defining data models, the NETCONF protocol was not usable for standards-based work. Existing data modeling languages such as the XML Schema Language (XSD) ([W3CXSD]) and the Document Schema Definition Languages (DSDL) ([ISODSDL]) were considered, but were rejected because the problem domains have little natural overlap. Defining a data model or protocol that is encoded in XML is a distinct problem from defining an XML document. The use of NETCONF operations place requirements on the data content that are not shared with the static document problem domain addressed by schema languages like XSD or RELAX NG.

In 2007 and 2008, the issue of a data modeling language for NETCONF was discussed in the OPS and APPS areas of IETF 70 and 71, and a design team was tasked with creating a requirements document (expired I-D draft-presuhn-rcdml-03.txt). After discussing the available options at the CANMOD BoF at IETF71, the community wrote a charter for the NETMOD working group. An excellent description of this time period is available at <http://www.ietf.org/mail-archive/web/ietf/current/msg51644.html>

In 2008 and 2009, the NETMOD working group produced a specification for YANG ([RFCYANG]) as a means for defining data models for NETCONF, allowing both standard and proprietary data models to be published in a form that is easily digestible by human readers and satisfies many of the issues raised in the IAB NM workshop. This brings NETCONF to a point where it can be used to develop standard data models within the IETF.

YANG allows a modeler to create a data model, to define the organization of the data in that model, and to define constraints on that data. Once published, the YANG module acts as a contract between the client and server, with both parties understanding how their peer will expect them to behave. A client knows how to create valid data for the server, and knows what data will be sent from the server. A server knows the rules that govern the data and how it should behave.

YANG also incorporates a level of extensibility and flexibility not present in other model languages. New modules can augment the data hierarchies defined in other modules, seamlessly adding data at appropriate places in the existing data organization. YANG also allows new statements to be defined, allowing the language itself to be expanded in a consistent way.

This document presents an architecture for YANG, describing how YANG-related technologies work and how solutions built on them can address the network management problem domain.

## 2. Elements of the Architecture

### 2.1. NETCONF

NETCONF defines an XML-based remote procedure call (RPC) mechanism that leverages the simplicity and availability of high-quality XML parsers. XML gives a rich, flexible, hierarchical, standard representation of data that matches the needs of networking devices. NETCONF carries configuration data and operations as requests and replies using RPCs encoded in XML over a connection-oriented transport.

XML's hierarchical data representation allows complex networking data to be rendered in a natural way. For example, the following configuration places interfaces in OSPF areas. The <ospf> element contains a list of <area> elements, each of which contain a list of <interface> elements. The <name> element identifies the specific area or interface. Additional configuration for each area or interface appears directly inside the appropriate element.

```
<ospf xmlns="http://example.org/netconf/ospf">
  <area>
    <name>0.0.0.0</name>

    <interface>
      <name>ge-0/0/0.0</name>
      <!-- The priority for this interface -->
      <priority>30</priority>
      <metric>100</metric>
      <dead-interval>120</dead-interval>
    </interface>

    <interface>
      <name>ge-0/0/1.0</name>
      <metric>140</metric>
    </interface>
  </area>

  <area>
    <name>10.1.2.0</name>

    <interface>
      <name>ge-0/0/2.0</name>
      <metric>100</metric>
    </interface>

    <interface>
      <name>ge-0/0/3.0</name>
      <metric>140</metric>
      <dead-interval>120</dead-interval>
    </interface>
  </area>
</ospf>
```

NETCONF includes mechanisms for controlling configuration datastores. Each datastore is a specific collection of configuration data that can be used as source or target of the configuration-related operations. The device can indicate whether it has a distinct "startup" configuration datastore, whether the current or "running" datastore is directly writable, or whether there is a "candidate" configuration datastore where configuration changes can be made that will not affect the device until a "commit-configuration" operation is invoked.

NETCONF defines operations that are invoked as RPCs from the client (the application) to the server (running on the device). The following table lists some of these operations:

Operation	Description
commit	Commits the "candidate" configuration to "running"
copy-config	Copy one configuration datastore to another
delete-config	Delete a configuration datastore
edit-config	Change the contents of a configuration datastore
get-config	Retrieve all or part of a configuration datastore
lock	Prevent changes to a datastore from another party
unlock	Release a lock on a datastore

NETCONF's "capability" mechanism allows the device to announce the set of capabilities that the device supports, including protocol operations, datastores, data models, and other abilities. These are announced during session establishment as part of the <hello> message. A client can inspect the hello message to determine what the device is capable of and how to interact with the device to perform the desired tasks.

NETCONF also defines a means of sending asynchronous notifications from the server to the client, described in [RFC5277].

In addition, NETCONF can fetch state data, receive notifications, and invoke additional RPC methods defined as part of a capability. Complete information about NETCONF can be found in [RFC4741].

#### 2.1.1. NETCONF Transport Mappings

NETCONF can run over any transport protocol that meets the requirements defined in RFC4741, including

- o connection-oriented operation
- o authentication
- o integrity
- o confidentiality

[RFC4742] defines an mapping for the SSH ([RFC4251]) protocol, which is the mandatory transport protocol. Others include SOAP ([RFC4743]), BEEP ([RFC4744]), and TLS ([RFC5539]).



## 2.2. YANG

YANG is a data modeling language for NETCONF. It allows the description of hierarchies of data nodes ("nodes") and the constraints that exist among them. YANG defines data models and how to manipulate those models via NETCONF protocol operations.

Each YANG module defines a data model, uniquely identified by a namespace URI. These data models are extensible in a manner that allows tight integration of standard data models and proprietary data models. Models are built from organizational containers, lists of data nodes and data node forming leafs of the data tree.

```

module example-ospf {
  namespace "http://example.org/netconf/ospf";
  prefix ospf;

  import network-types { // Access another module's def'ns
    prefix nett;
  }

  container ospf { // Declare the top-level tag
    list area { // Declare a list of "area" nodes
      key name; // The key "name" identifies list members
      leaf name {
        type nett:area-id;
      }
      list interface {
        key name;
        leaf name {
          type nett:interface-name;
        }
        leaf priority {
          description "Designated router priority";
          type uint8; // The type is a constraint on
                      // valid values for "priority".
        }
        leaf metric {
          type uint16 {
            range 1..65535;
          }
        }
        leaf dead-interval {
          units seconds;
          type uint16 {
            range 1..65535;
          }
        }
      }
    }
  }
}

```

A YANG module defines a data model in terms of the data, its hierarchical organization, and the constraints on that data. YANG defines how this data is represented in XML and how that data is used in NETCONF operations.

The following table briefly describes some common YANG statements:

Statement	Description
augment	Extends existing data hierarchies
choice	Defines mutually exclusive alternatives
container	Defines a layer of the data hierarchy
extension	Allows new statements to be added to YANG
feature	Indicates parts of the model are optional
grouping	Groups data definitions into reusable sets
key	Defines the key leafs for lists
leaf	Defines a leaf node in the data hierarchy
leaf-list	A leaf node that can appear multiple times
list	A hierarchy that can appear multiple times
notification	Defines notification
rpc	Defines input and output parameters for an RPC operation
typedef	Defines a new type
uses	Incorporates the contents of a "grouping"

### 2.2.1. Constraints

YANG allows the modeler to add constraints to the data model to prevent impossible or illogical data. These constraints give clients information about the data being sent from the device, and also allow the client to know as much as possible about the data the device will accept, so the client can send correct data. These constraints apply to configuration data, but can also be used for rpc and notification data.

The principal constraint is the "type" statement, which limits the contents of a leaf node to that of the named type. The following table briefly describes some other common YANG constraints:

Statement	Description
length	Limits the length of a string
mandatory	Requires the node appear
max-elements	Limits the number of instances in a list
min-elements	Limits the number of instances in a list
must	XPath expression must be true
pattern	Regular expression must be satisfied
range	Value must appear in range
reference	Value must appear elsewhere in the data
unique	Value must be unique within the data
when	Node is only present when XPath expression is true

The "must" and "when" statements use XPath ([W3CXPATH]) expressions to specify conditions that are semantically evaluated against the data hierarchy, but neither the client nor the server are required to implement the XPath specification. Instead they can use any means to ensure these conditions are met.

#### 2.2.2. Flexibility

YANG uses the "union" type and the "choice" and "feature" statements to give modelers flexibility in defining their data models. The "union" type allows a single leaf to accept multiple types, like an integer or the word "unbounded":

```
type union {  
    type int32;  
    type enumeration {  
        enum "unbounded";  
    }  
}
```

The "choice" statement lists a set of mutually exclusive nodes, so a valid configuration can choose any one node (or case). The "feature" statement allows the modeler to identify parts of the model which can be optional, and allows the device to indicate whether it implements these optional portions.

The "deviation" statement allows the device, to indicate parts of a YANG module which the device does not faithfully implement. While devices are encouraged to fully abide according to the contract presented in the YANG module, real world situations may force the device to break the contract. Deviations give a means of declaring this limitation, rather than leaving it to be discovered via run-time errors.

#### 2.2.3. Extensibility Model

XML includes the concept of namespaces, allowing XML elements from different sources to be combined in the same hierarchy without risking collision. YANG modules define content for specific namespaces, but one module may augment the definition of another module, introducing elements from that module's namespace into the first module's hierarchy.

Since one module can augment another module's definition, hierarchies of definitions are allowed to grow, as definitions from multiple sources are added to the base hierarchy. These augmentations are qualified using the namespace of the source module, helping to avoid issues with name conflicts as the modules change over time.

For example, if the above OSPF configuration were the standard, a vendor module may augment this with vendor-specific extensions.

```
module vendorx-ospf {
  namespace "http://vendorx.example.com/ospf";
  prefix vendorx;

  import example-ospf {
    prefix ospf;
  }

  augment /ospf:ospf/ospf:area/ospf:interfaces {
    leaf no-neighbor-down-notification {
      type empty;
      description "Don't inform other protocols about "
        + " neighbor down events";
    }
  }
}
```

The <no-neighbor-down-notification> element is then placed in the vendorx namespace:

```
<ospf xmlns="http://example.org/netconf/ospf"
      xmlns:vendorx="http://vendorx.example.com/ospf">

  <area>
    <name>0.0.0.0</name>

    <interface>
      <name>ge-0/0/0.0</name>
      <priority>30</priority>
      <vendorx:no-neighbor-down-notification/>
    </interface>

  </area>
</ospf>
```

Augmentations are seamlessly integrated with base modules, allowing them to be fetched, archived, loaded, and deleted within their natural hierarchy. If a client application asks for the configuration for a specific OSPF area, it will receive the sub-hierarchy for that area, complete with any augmented data.

### 2.3. YANG Translations

The YANG data modeling language is the central piece of a group of related technologies. The YANG language itself, described in

[RFCYANG], defines the syntax of the language and its statements, the meaning of those statements, and how to combine them to build the hierarchy of nodes that describe a data model.

That document also defines the "on the wire" XML content for NETCONF operations on data models defined in YANG modules. This includes the basic mapping between YANG data tree nodes and XML elements, as well as mechanisms used in <edit-config> content to manipulate that data, such as arranging the order of nodes within a list.

YANG uses a syntax that is regular and easily described, primarily designed for human readability. YANG's syntax is friendly to email, diff, patch, and the constraints of RFC formatting.

#### 2.3.1. YIN

In some environments, incorporating a YANG parser may not be an acceptable option. For those scenarios, an XML grammar for YANG is defined as YIN (YANG Independent Notation). YIN allows the use of XML parsers which are readily available in both open source and commercial versions. Conversion between YANG and YIN is direct, loss-less and reversible. YANG statements are converted to XML elements, preserving the structure and content of YANG, but enabling the use of off-the-shelf XML parsers rather than requiring the integration of a YANG parser. YIN maintains complete semantic equivalence with YANG.

#### 2.3.2. DSDL (RELAX NG)

Since NETCONF content is encoded in XML, it is natural to use XML schema languages for their validation. To facilitate this, YANG offers a standardized mapping of YANG modules into Document Schema Description Languages ([RFCYANGDSDL]), of which RELAX NG is a major component.

DSDL is considered to be the best choice as a standard schema language because it addresses not only grammar and datatypes of XML documents but also semantic constraints and rules for modifying the information set of the document.

In addition, DSDL offers formal means for coordinating multiple independent schemas and specifying how to apply the schemas to the various parts of the document. This is useful since YANG content is typically composed of multiple vocabularies.

## 2.4. YANG Types

YANG supports a number of builtin types, and allows additional types to be derived from those types in an extensible manner. New types can add additional restrictions to allowable data values.

A standard type library for use by YANG is available [RFCYANGTYPES]. These YANG modules define commonly used data types for IETF-related standards.

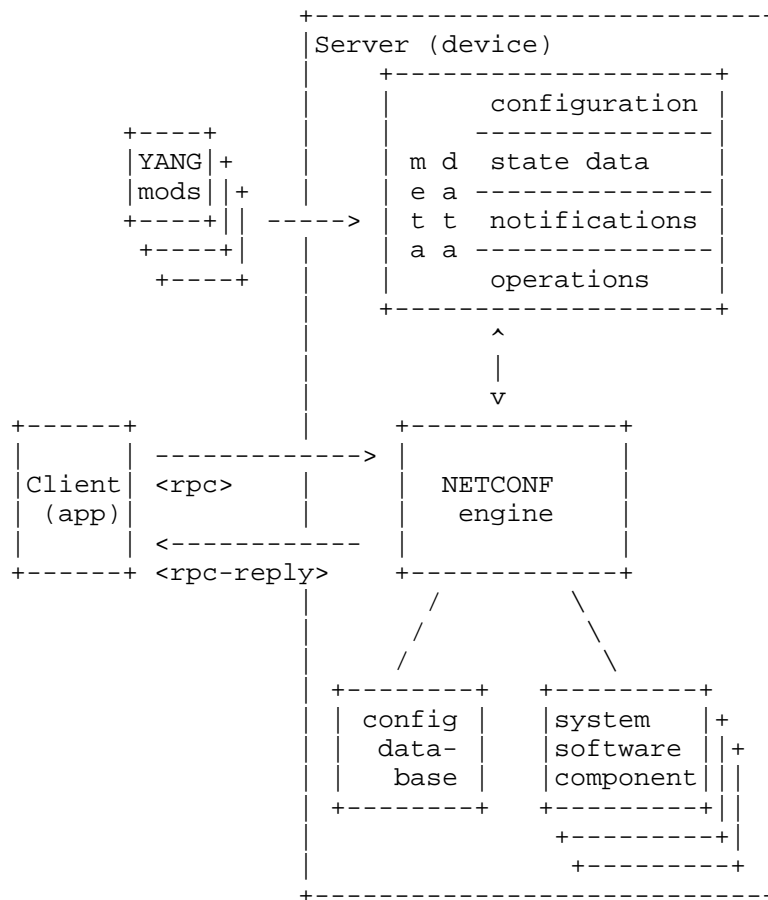
## 2.5. IETF Guidelines

A set of additional guidelines are defined that indicate desirable usage for authors and reviewers of standards track specifications containing YANG data model modules ([RFCYANGUSAGE]). These guidelines should be used as a basis for reviews of other YANG data model documents.

### 3. Working with YANG

#### 3.1. Building NETCONF- and YANG-based Solutions

In the typical YANG-based solution, the client and server are driven by the content of YANG modules. The server includes the definitions of the modules as meta-data that is available to the NETCONF engine. This engine processes incoming requests, uses the meta-data to parse and verify the request, performs the requested operation, and returns the results to the client.



To use YANG, YANG modules must be defined to model the specific problem domain. These modules are then loaded, compiled, or coded into the server.

The sequence of events for the typical client/server interaction may



be as follows:

- o A client application ([C]) opens a NETCONF session to the server (device) ([S])
- o [C] and [S] exchange <hello> messages containing the list of capabilities supported by each side, allowing [C] to learn the modules supported by [S]
- o [C] builds and sends an operation defined in the YANG module, encoded in XML, within NETCONF's <rpc> element
- o [S] receives and parses the <rpc> element
- o [S] verifies the contents of the request against the data model defined in the YANG module
- o [S] performs the requested operation, possibly changing the configuration datastore
- o [S] builds the response, containing the response, any requested data, and any errors
- o [S] sends the response, encoded in XML, within NETCONF's <rpc-reply> element
- o [C] receives and parses the <rpc-reply> element
- o [C] inspects the response and processes it as needed

Note that there is no requirement for the client or server to process the YANG modules in this way. The server may hard code the contents of the data model, rather than handle the content via a generic engine. Or the client may be targeted at the specific YANG model, rather than being driven generically. Such a client might be a simple shell script that stuffs arguments into an XML payload template and sends it to the server.

### 3.2. Addressing Operator Requirements

NETCONF and YANG address many of the issues raised in the IAB NM workshop.

- o Ease of use: YANG is designed to be human friendly, simple and readable. Many tricky issues remain due to the complexity of the problem domain, but YANG strives to make them more visible and easier to deal with.

- o Configuration and state data: YANG clearly divides configuration data from other types of data.
- o Transactions: NETCONF provides a simple transaction mechanism.
- o Generation of deltas: A YANG module gives enough information to generate the delta needed to change between two configuration data sets.
- o Dump and restore: NETCONF gives the ability to save and restore configuration data. This can also be performed for a specific YANG module.
- o Network-wide configuration: NETCONF supports robust network-wide configuration transactions via the commit and confirmed-commit capability. When a change is attempted that affects multiple devices, these capabilities simplify the management of failure scenarios, resulting in the ability to have transactions that will dependably succeed or fail atomically.
- o Text-friendly: YANG modules are very text friendly, as is the data they define.
- o Configuration handling: NETCONF addresses the ability to distinguish between distributing configuration data and activating it.
- o Task-oriented: A YANG module can define specific tasks as RPC operations. A client can choose to invoke the RPC operation or to access any underlying data directly.
- o Full coverage: YANG modules can be defined that give full coverage to all the native abilities of the device. Giving this access avoids the need to resort to the command line interface (CLI) using tools such as Expect ([SWEXPECT]).
- o Timeliness: YANG modules can be tied to CLI operations, so all native operations and data are immediately available.
- o Implementation difficulty: YANG's flexibility enables modules that can be more easily implemented. Adding "features" and replacing "third normal form" with a natural data hierarchy should reduce complexity.
- o Simple data modeling language: YANG has sufficient power to be usable in other situations. In particular, on-box API and native CLI can be integrated to achieve simplification of the infrastructure.

- o Internationalization: YANG uses UTF-8 ([RFC3629]) encoded unicode characters.
- o Event correlation: YANG integrates RPC operations, notification, configuration and state data, enabling internal references. For example, a field in a notification can be tagged as pointing to a BGP peer, and the client application can easily find that peer in the configuration data.
- o Implementation costs: Significant effort has been made to keep implementation costs as low as possible.
- o Human friendly syntax: YANG's syntax is optimized for the reader, specifically the reviewer on the basis that this is the most common human interaction.
- o Post-processing: Use of XML will maximize the opportunities for post-processing of data, possibly using XML-based technologies like XPath ([W3CXPATH], XQuery ([W3CXQUERY]), and XSLT ([W3CXSLT]).
- o Semantic mismatch: Richer, more descriptive data models will reduce the possibility of semantic mismatch. With the ability to define new primitives, YANG modules will be more specific in content, allowing more enforcement of rules and constraints.
- o Security: NETCONF runs over transport protocols secured by SSH or TLS, allowing secure communications and authentication using well-trusted technology. The secure transport can use existing key and credential management infrastructure, reducing deployment costs.
- o Reliable: NETCONF and YANG are solid and reliable technologies. NETCONF is connection based, and includes automatic recovery mechanisms when the connection is lost.
- o Delta friendly: YANG-based models support operations that are delta friendly. Add, change, insert, and delete operations are all well defined.
- o Method-oriented: YANG allows new RPC operations to be defined, including an operation name, which is essentially a method. The input and output parameters of the RPC operations are also defined in the YANG module.

### 3.3. Roles in Building Solutions

Building NETCONF- and YANG-based solutions requires interacting with many distinct groups. Modelers must understand how to build useful models that give structure and meaning to data while maximizing the flexibility of that data to "future proof" their work. Reviewers need to quickly determine if that structure is accurate. Device developers need to code that data model into their devices, and application developers need to code their applications to take advantage of that data model. There are a variety of strategies for performing each piece of this work. This section discusses some of those strategies.

#### 3.3.1. Modeler

The modeler defines a data model based on their in-depth knowledge of the problem domain being modeled. This model should be as simple as possible, but should balance complexity with expressiveness. The organization of the model should target not only the current model, but should allow for extensibility from other modules and for adaptability to future changes.

Additional modeling issues are discussed in Section 4.

#### 3.3.2. Reviewer

The reviewer role is perhaps the most important and the time reviewers are willing to give is precious. To help the reviewer, YANG stresses readability, with a human-friendly syntax, natural data hierarchy, and simple, concise statements.

#### 3.3.3. Device Developer

The YANG model tells the device developer what data is being modeled. The developer reads the YANG models and writes code that supports the model. The model describes the data hierarchy and associated constraints, and the description and reference material helps the developer understand how to transform the models view into the device's native implementation.

##### 3.3.3.1. Generic Content Support

The YANG model can be compiled into a YANG-based engine for either the client or server side. Incoming data can be validated, as can outgoing data. The complete configuration datastore may be validated in accordance with the constraints described in the data model.

Serializers and deserializers for generating and receiving NETCONF

content can be driven by the meta-data in the model. As data is received, the meta-data is consulted to ensure the validity of incoming XML elements.

#### 3.3.3.2. XML Definitions

The YANG module dictates the XML encoding for data sent via NETCONF. The rules that define the encoding are fixed, so the YANG module can be used to ascertain whether a specific NETCONF payload is obeying the rules.

#### 3.3.4. Application Developer

The YANG module tells the application developer what data can be modeled. Developers can inspect the modules and take one of three distinct views. In this section, we will consider them and the impact of YANG on their design. In the real world, most applications are a mixture of these approaches.

##### 3.3.4.1. Hard Coded

An application can be coded against the specific, well-known contents of YANG modules, implementing their organization, rules, and logic directly with explicit knowledge. For example, a script could be written to change the domain name of a set of devices using a standard YANG module that includes such a leaf node. This script takes the new domain name as an argument and inserts it into a string containing the rest of the XML encoding as required by the YANG module. This content is then sent via NETCONF to each of the devices.

This type of application is useful for small, fixed problems where the cost and complexity of flexibility is overwhelmed by the ease of hard coding direct knowledge into the application.

##### 3.3.4.2. Bottom Up

An application may take a generic, bottom up approach to configuration, concentrating on the device's data directly and treating that data without specific understanding.

YANG modules may be used to drive the operation of the YANG equivalent of a "MIB Browser". Such an application manipulates the device's configuration data based on the data organization contained in the YANG module. For example, a GUI may present a straightforward visualization where elements of the YANG hierarchy are depicted in a hierarchy of folders or GUI panels. Clicking on a line expands to the contents of the matching XML hierarchy.

This type of GUI can easily be built by generating XSLT stylesheets from the YANG data models. An XSLT engine can then be used to turn configuration data into a set of web pages.

The YANG modules allow the application to enforce a set of constraints without understanding the semantics of the YANG module.

#### 3.3.4.3. Top Down

In contrast to the bottom-up approach, the top-down approach allows the application to take a view of the configuration data which is distinct from the standard and/or proprietary YANG modules. The application is free to construct its own model for data organization and to present this model to the user. When the application needs to transmit data to a device, the application transforms its data from the problem-oriented view of the world into the data needed for that particular device. This transformation is under the control and maintenance of the application, allowing the transformation to be changed and updated without affecting the device.

For example, an application could be written that models VPNs in a network-oriented view. The application would need to transform these high-level VPN definitions into the configuration data that would be handed to any particular device within a VPN.

Even in this approach, YANG is useful since it can be used to model the VPN. For example, the following VPN straw-man models a list of VPNs, each with a protocol, a topology, a list of member interfaces, and a list of classifiers.

```
list example-bgpvpn {
  key name;
  leaf name { ... }
  leaf protocol {
    type enumeration {
      enum bgpvpn;
      enum l2vpn;
    }
  }
  leaf topology {
    type enumeration {
      enum hub-n-spoke;
      enum mesh;
    }
  }
  list members {
    key "device interface";
    leaf device { ... }
    leaf interface { ... }
  }
  list classifiers {
    ...
  }
}
```

The application can use such a YANG module to drive its operation, building VPN instances in a database and then pushing the configuration for those VPNs to individual devices using either a standard device model (e.g. example-bgpvpn.yang) or by transforming that standard device content into some proprietary format for devices that do not support that standard.

#### 4. Modeling Considerations

This section discusses considerations the modeler should be aware of while developing models in YANG.

##### 4.1. Default Values

The concept of default values is simple, but their details, representation, and interaction with configuration data can be difficult issues. NETCONF leaves default values as a data model issue, and YANG gives flexibility to the device implementation in terms of how default values are handled. The requirement is that the device "MUST operationally behave as if the leaf was present in the data tree with the default value as its value". This gives the device implementation choices in how default values are handled.

One choice is to view the configuration as a set of instructions for how the device should be configured. If a data value that is given as part of those instructions is the default value, then it should be retained as part of the configuration, but if it is not explicitly given, then the value is not considered to be part of configuration.

Another choice is to trim values that are identical to the default values, implicitly removing them from the configuration datastore. The act of setting a leaf to its default value effectively deletes that leaf.

The device could also choose to report all default values, regardless of whether they were explicitly set. This choice eases the work of a client that needs default values, but may significantly increase the size of the configuration data.

These choices reflect the default handling schemes of widely deployed networking devices and supporting them allows YANG to reduce implementation and deployment costs of YANG-based models.

When the client retrieves data from the device, it must be prepared to handle the absence of leaf nodes with the default value, since the server is not required to send such leaf elements. This permits the device to implement either of the first two default handling schemes given above.

Regardless of the implementation choice, the device can support the "with-defaults" capability ([RFCWITHDEFAULTS]) and give the client the ability to select the desired handling of default values.

When evaluating the XPath expressions for constraints like "must" and "when", the evaluation context for the expressions will include any



appropriate default values, so the modeler can depend on consistent behavior from all devices.

#### 4.2. Compliance

In developing good data models, there are many conflicting interests the data modeler must keep in mind. Modelers need to be aware of five issues with models and devices:

- o usefulness
- o compliance
- o flexibility
- o extensibility
- o deviations

For a model to be interesting, it must be useful, solving a problem in a more direct or more powerful way than can be accomplished without the model. The model should maximize the usefulness of the model within the problem domain.

Modelers should build models that maximize the number of devices that can faithfully implement the model. If the model is drawn too narrowly, or includes too many assumptions about the device, then the difficulty and cost of accurately implementing the model will lead to low quality implementations, interoperability issues, and will reduce the value of the model.

Modelers can use the "feature" statement in their models to give the device some flexibility by partitioning their model and allowing the device to indicate which portions of the model are implemented on the device. For example, if the model includes some a "logging" feature, a device with no storage facilities for the log can tell the client that it does not support this feature of the model.

Models can be extended via the "augment" statement, and the modeler should consider how their model is likely to be extended. These augmentations can be defined by vendors, applications, or standards bodies.

Deviations are a means of allowing the devices to indicate where its implementation is not in full compliance with the model. For example, once a model is published, an implementer may decide to make a particular node configurable, where the standard model describes it as state data. The implementation reports the value normally and may

declare a deviation that this device behaves in a different manner than the standard. Applications capable of discovering this deviation can make allowances, but applications that do not discover the deviation can continue treating the implementation as if it were compliant.

Rarely, implementations may make decisions that prevent compliance with the standard. Such occasions are regrettable, but they remain a part of reality, and modelers and application writers ignore them at their own risk. An implementation that emits an integer leaf as "cow" would be difficult to manage, but applications should expect to encounter such misbehaving devices in the field.

Despite this, both client and server should view the YANG module as a contract, with both sides agreeing to abide by the terms. The modeler should be explicit about the terms of such a contract, and both client and server implementations should strive to faithfully and accurately implement the data model described in the YANG module.

#### 4.3. Data Distinctions

The distinction between configuration data, operational state data, and statistics is important to understand for data model writers and people who plan to extend the NETCONF protocol. This section first discusses some background and then provides a definition and some examples.

##### 4.3.1. Background

During the IAB NM workshop, operators did formulate the following two requirements:

2. It is necessary to make a clear distinction between configuration data, data that describes operational state and statistics. Some devices make it very hard to determine which parameters were administratively configured and which were obtained via other mechanisms such as routing protocols.
3. It is required to be able to fetch separately configuration data, operational state data, and statistics from devices, and to be able to compare these between devices.

The NETCONF protocol defined in RFC 4741 distinguishes two types of data, namely configuration data and state data:

Configuration data is the set of writable data that is required to transform a system from its initial default state into its current state.

State data is the additional data on a system that is not configuration data such as read-only status information and collected statistics.

NETCONF does not follow the distinction formulated by the operators between configuration data, operational state data, and statistical data, since it considers state data to include both statistics and operational state data.

#### 4.3.2. Definitions

Below is a definition for configuration data, operational state data, and statistical data. The definition borrows from previous work.

- o Configuration data is the set of writable data that is required to transform a system from its initial default state into its current state. [RFC4741]
- o Operational state data is a set of data that has been obtained by the system at runtime and influences the system's behaviour similar to configuration data. In contrast to configuration data, operational state is transient and modified by interactions with internal components or other systems via specialized protocols.
- o Statistical data is the set of read-only data created by a system itself. It describes the performance of the system and its components.

The following examples help to clarify the difference between configuration data, operational state data and statistical data.

##### 4.3.2.1. Example 1: IP Routing Table

IP routing tables can contain entries that are statically configured (configuration data) as well as entries obtained from routing protocols such as OSPF (operational state data). In addition, a routing engine might collect statistics like how often a particular routing table entry has been used.

##### 4.3.2.2. Example 2: Interfaces

Network interfaces usually come with a large number of attributes that are specific to the interface type and in some cases specific to the cable plugged into an interface. Examples are the maximum

transmission unit of an interface or the speed detected by an Ethernet interface.

In many deployments, systems use the interface attributes detected when an interface is initialized. As such, these attributes constitute operational state. However, there are usually provisions to overwrite the discovered attributes with static configuration data, like for example configuring the interface MTU to use a specific value or forcing an Ethernet interface to run at a given speed.

The system will record statistics (counters) measuring the number of packets, bytes, and errors received and transmitted on each interface.

#### 4.3.2.3. Example 3: Account Information

Systems usually maintain static configuration information about the accounts on the system. In addition, systems can obtain information about accounts from other sources (e.g. LDAP, NIS) dynamically, leading to operational state data. Information about account usage are examples of statistic data.

Note that configuration data supplied to a system in order to create a new account might be supplemented with additional configuration information determined by the system when the account is being created (such as a unique account id). Even though the system might create such information, it usually becomes part of the static configuration of the system since this data is not transient.

#### 4.3.3. Implications

The primary focus of YANG is configuration data. There is no single mechanism defined for the separation of operational state data and statistics since NETCONF treats them both as state data. This section describes several different options for addressing this issue.

##### 4.3.3.1. Data Models

The first option is to have data models that explicitly differentiate between configuration data and operational state data. This leads to duplication of data structures and might not scale well from a modeling perspective.

For example, the configured duplex value and the operational duplex value would be distinct leafs in the data model.

#### 4.3.3.2. Additional Operations to Retrieve Operational State

The NETCONF protocol can be extended with new protocol operations that specifically allow the retrieval of all operational state, e.g. by introducing a <get-ops> operation (and perhaps also a <get-stats> operation).

#### 4.3.3.3. Introduction of an Operational State Datastore

Another option could be to introduce a new "configuration" data store that represents the operational state. A <get-config> operation on the <operational> data store would then return the operational state determining the behaviour of the box instead of its static and explicit configuration state.

#### 4.4. Direction

At this time, the only viable solution is to distinctly model the configuration and operational values. The configuration leaf would indicate the desired value, as given by the user, and the operational leaf would indicate the current value, as observed on the device.

In the duplex example, this would result in two distinct leafs being defined, "duplex" and "op-duplex", one with "config true" and one with "config false".

In some cases, distinct leafs would be used, but in others, distinct lists might be used. Distinct lists allows the list to be organized in different ways, with different constraints. Keys, sorting, and constraint statements like must, unique, or when may differ between configuration data and operational data.

For example, configured static routes might be a distinct list from the operational routing table, since the use of keys and sorting might differ.

## 5. Security Considerations

This document discusses an architecture for network management using NETCONF and YANG. It has no security impact on the Internet.

## 6. IANA Considerations

This document has no actions for IANA.

## 7. Normative References

- [ISODSDL] International Organization for Standardization, "Document Schema Definition Languages (DSDL) - Part 1: Overview", ISO/IEC 19757-1, November 2004.
- [RFC3535] Schoenwaelder, J., "Overview of the 2002 IAB Network Management Workshop", RFC 3535, May 2003.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.
- [RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.
- [RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.
- [RFC4742] Wasserman, M. and T. Goddard, "Using the NETCONF Configuration Protocol over Secure SHell (SSH)", RFC 4742, December 2006.
- [RFC4743] Goddard, T., "Using NETCONF over the Simple Object Access Protocol (SOAP)", RFC 4743, December 2006.
- [RFC4744] Lear, E. and K. Crozier, "Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)", RFC 4744, December 2006.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.
- [RFC5539] Badra, M., "NETCONF over Transport Layer Security (TLS)", RFC 5539, May 2009.
- [RFCWITHDEFAULTS] Bierman, A. and B. Lengyel, "With-defaults capability for NETCONF", draft-ietf-netconf-with-defaults-11.txt (work in progress).
- [RFCYANG] Bjorklund, M., Ed., "YANG - A data modeling language for the Network Configuration Protocol (NETCONF)", draft-ietf-netmod-yang-13 (work in progress).
- [RFCYANGDSDL] Lhotka, L., Mahy, R., and S. Chishom, "Mapping YANG to Document Schema Definition Languages and Validating NETCONF Content", draft-ietf-netmod-dsdl-map-07 (work in progress).



progress).

[RFCYANGTYPES]

Schoenwaelder, J., "Common YANG Data Types",  
draft-ietf-netmod-yang-types-09.txt (work in progress).

[RFCYANGUSAGE]

Bierman, A., "Guidelines for Authors and Reviewers of YANG  
Data Model Documents", draft-ietf-netmod-yang-usage-10.txt  
(work in progress).

[SWEXPECT]

"The Expect Home Page", <<http://expect.sourceforge.net/>>.

[W3CXPATH]

DeRose, S. and J. Clark, "XML Path Language (XPath)  
Version 1.0", World Wide Web Consortium  
Recommendation REC-xpath-19991116, November 1999,  
<<http://www.w3.org/TR/1999/REC-xpath-19991116>>.

[W3CXQUERY]

Boag, S., "XQuery 1.0: An XML Query Language", W3C WD WD-  
xquery-20050915, September 2005.

[W3CXSD]

Walmsley, P. and D. Fallside, "XML Schema Part 0: Primer  
Second Edition", World Wide Web Consortium  
Recommendation REC-xmlschema-0-20041028, October 2004,  
<<http://www.w3.org/TR/2004/REC-xmlschema-0-20041028>>.

[W3CXSLT]

Clark, J., "XSL Transformations (XSLT) Version 1.0", World  
Wide Web Consortium Recommendation REC-xslt-19991116,  
November 1999,  
<<http://www.w3.org/TR/1999/REC-xslt-19991116>>.

Author's Address

Phil Shafer  
Juniper Networks

Email: [phil@juniper.net](mailto:phil@juniper.net)



NETMOD  
Internet-Draft  
Intended status: Standards Track  
Expires: April 24, 2011

L. Lhotka, Ed.  
CESNET  
October 21, 2010

Mapping YANG to Document Schema Definition Languages and Validating  
NETCONF Content  
draft-ietf-netmod-dsdl-map-10

Abstract

This document specifies the mapping rules for translating YANG data models into Document Schema Definition Languages (DSDL), a coordinated set of XML schema languages standardized as ISO/IEC 19757. The following DSDL schema languages are addressed by the mapping: RELAX NG, Schematron and DSRL. The mapping takes one or more YANG modules and produces a set of DSDL schemas for a selected target document type - datastore content, NETCONF message etc. Procedures for schema-based validation of such documents are also discussed.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 24, 2011.

Copyright Notice

Copyright (c) 2010 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.

## Table of Contents

1. Introduction . . . . .	6
2. Terminology and Notation . . . . .	8
2.1. Glossary of New Terms . . . . .	11
3. Objectives and Motivation . . . . .	12
4. DSDL Schema Languages . . . . .	14
4.1. RELAX NG . . . . .	14
4.2. Schematron . . . . .	15
4.3. Document Semantics Renaming Language (DSRL) . . . . .	16
5. Additional Annotations . . . . .	17
5.1. Dublin Core Metadata Elements . . . . .	17
5.2. RELAX NG DTD Compatibility Annotations . . . . .	17
5.3. NETMOD-Specific Annotations . . . . .	18
6. Overview of the Mapping . . . . .	20
7. NETCONF Content Validation . . . . .	22
8. Design Considerations . . . . .	23
8.1. Hybrid Schema . . . . .	23
8.2. Modularity . . . . .	25
8.3. Granularity . . . . .	27
8.4. Handling of XML Namespaces . . . . .	27
9. Mapping YANG Data Models to the Hybrid Schema . . . . .	29
9.1. Occurrence Rules for Data Nodes . . . . .	29
9.1.1. Optional and Mandatory Nodes . . . . .	30
9.1.2. Implicit Nodes . . . . .	31
9.2. Mapping YANG Groupings and Typedefs . . . . .	32
9.2.1. YANG Refinements and Augments . . . . .	33
9.2.2. Type Derivation Chains . . . . .	36
9.3. Translation of XPath Expressions . . . . .	38
9.4. YANG Language Extensions . . . . .	39
10. Mapping YANG Statements to the Hybrid Schema . . . . .	41
10.1. The 'anyxml' Statement . . . . .	41
10.2. The 'argument' Statement . . . . .	42
10.3. The 'augment' Statement . . . . .	43
10.4. The 'base' Statement . . . . .	43
10.5. The 'belongs-to' Statement . . . . .	43
10.6. The 'bit' Statement . . . . .	43
10.7. The 'case' Statement . . . . .	43
10.8. The 'choice' Statement . . . . .	43
10.9. The 'config' Statement . . . . .	44
10.10. The 'contact' Statement . . . . .	44

10.11.	The 'container' Statement . . . . .	44
10.12.	The 'default' Statement . . . . .	44
10.13.	The 'description' Statement . . . . .	46
10.14.	The 'deviation' Statement . . . . .	46
10.15.	The 'enum' Statement . . . . .	46
10.16.	The 'error-app-tag' Statement . . . . .	46
10.17.	The 'error-message' Statement . . . . .	46
10.18.	The 'extension' Statement . . . . .	46
10.19.	The 'feature' Statement . . . . .	46
10.20.	The 'grouping' Statement . . . . .	46
10.21.	The 'identity' Statement . . . . .	47
10.22.	The 'if-feature' Statement . . . . .	48
10.23.	The 'import' Statement . . . . .	49
10.24.	The 'include' Statement . . . . .	49
10.25.	The 'input' Statement . . . . .	49
10.26.	The 'key' Statement . . . . .	49
10.27.	The 'leaf' Statement . . . . .	49
10.28.	The 'leaf-list' Statement . . . . .	50
10.29.	The 'length' Statement . . . . .	50
10.30.	The 'list' Statement . . . . .	51
10.31.	The 'mandatory' Statement . . . . .	52
10.32.	The 'max-elements' Statement . . . . .	52
10.33.	The 'min-elements' Statement . . . . .	52
10.34.	The 'module' Statement . . . . .	52
10.35.	The 'must' Statement . . . . .	53
10.36.	The 'namespace' Statement . . . . .	53
10.37.	The 'notification' Statement . . . . .	54
10.38.	The 'ordered-by' Statement . . . . .	54
10.39.	The 'organization' Statement . . . . .	54
10.40.	The 'output' Statement . . . . .	54
10.41.	The 'path' Statement . . . . .	54
10.42.	The 'pattern' Statement . . . . .	54
10.43.	The 'position' Statement . . . . .	55
10.44.	The 'prefix' Statement . . . . .	55
10.45.	The 'presence' Statement . . . . .	55
10.46.	The 'range' Statement . . . . .	55
10.47.	The 'reference' Statement . . . . .	55
10.48.	The 'require-instance' Statement . . . . .	55
10.49.	The 'revision' Statement . . . . .	55
10.50.	The 'rpc' Statement . . . . .	55
10.51.	The 'status' Statement . . . . .	56
10.52.	The 'submodule' Statement . . . . .	56
10.53.	The 'type' Statement . . . . .	56
10.53.1.	The "empty" Type . . . . .	57
10.53.2.	The "boolean" Type . . . . .	57
10.53.3.	The "binary" Type . . . . .	58
10.53.4.	The "bits" Type . . . . .	58
10.53.5.	The "enumeration" and "union" Types . . . . .	58

10.53.6.	The "identityref" Type . . . . .	58
10.53.7.	The "instance-identifier" Type . . . . .	59
10.53.8.	The "leafref" Type . . . . .	59
10.53.9.	The Numeric Types . . . . .	59
10.53.10.	The "string" Type . . . . .	61
10.53.11.	Derived Types . . . . .	62
10.54.	The 'typedef' Statement . . . . .	63
10.55.	The 'unique' Statement . . . . .	63
10.56.	The 'units' Statement . . . . .	64
10.57.	The 'uses' Statement . . . . .	64
10.58.	The 'value' Statement . . . . .	64
10.59.	The 'when' Statement . . . . .	64
10.60.	The 'yang-version' Statement . . . . .	64
10.61.	The 'yin-element' Statement . . . . .	64
11.	Mapping the Hybrid Schema to DSDL . . . . .	65
11.1.	Generating RELAX NG Schemas for Various Document Types . . . . .	65
11.2.	Mapping Semantic Constraints to Schematron . . . . .	66
11.2.1.	Constraints on Mandatory Choice . . . . .	69
11.3.	Mapping Default Values to DSRL . . . . .	70
12.	Mapping NETMOD-specific Annotations to DSDL Schema Languages . . . . .	75
12.1.	The @nma:config Annotation . . . . .	75
12.2.	The @nma:default Annotation . . . . .	75
12.3.	The <nma:error-app-tag> Annotation . . . . .	75
12.4.	The <nma:error-message> Annotation . . . . .	75
12.5.	The @if-feature Annotation . . . . .	75
12.6.	The @nma:implicit Annotation . . . . .	76
12.7.	The <nma:instance-identifier> Annotation . . . . .	76
12.8.	The @nma:key Annotation . . . . .	76
12.9.	The @nma:leaf-list Annotation . . . . .	76
12.10.	The @nma:leafref Annotation . . . . .	77
12.11.	The @nma:min-elements Annotation . . . . .	77
12.12.	The @nma:max-elements Annotation . . . . .	77
12.13.	The <nma:must> Annotation . . . . .	77
12.14.	The <nma:ordered-by> Annotation . . . . .	78
12.15.	The <nma:status> Annotation . . . . .	78
12.16.	The @nma:unique Annotation . . . . .	78
12.17.	The @nma:when Annotation . . . . .	78
13.	IANA Considerations . . . . .	79
14.	Security Considerations . . . . .	80
15.	Contributors . . . . .	81
16.	Acknowledgments . . . . .	82
17.	References . . . . .	83
17.1.	Normative References . . . . .	83
17.2.	Informative References . . . . .	84
Appendix A.	RELAX NG Schema for NETMOD-Specific Annotations . . . . .	86
Appendix B.	Schema-Independent Library . . . . .	91
Appendix C.	Mapping DHCP Data Model - A Complete Example . . . . .	92

C.1.	Input YANG Module . . . . .	92
C.2.	Hybrid Schema . . . . .	94
C.3.	Final DSDL Schemas . . . . .	99
C.3.1.	Main RELAX NG Schema for <nc:get> Reply . . . . .	100
C.3.2.	RELAX NG Schema - Global Named Pattern Definitions . . . . .	102
C.3.3.	Schematron Schema for <nc:get> Reply . . . . .	104
C.3.4.	DSRL Schema for <nc:get> Reply . . . . .	106
Appendix D.	Change Log . . . . .	107
D.1.	Changes Between Versions -07 and -08 . . . . .	107
D.2.	Changes Between Versions -06 and -07 . . . . .	107
D.3.	Changes Between Versions -05 and -06 . . . . .	107
D.4.	Changes Between Versions -04 and -05 . . . . .	108
D.5.	Changes Between Versions -03 and -04 . . . . .	108
D.6.	Changes Between Versions -02 and -03 . . . . .	109
D.7.	Changes Between Versions -01 and -02 . . . . .	110
D.8.	Changes Between Versions -00 and -01 . . . . .	110
Author's	Address . . . . .	112



## 1. Introduction

The NETCONF Working Group has completed a base protocol used for configuration management [RFC4741]. This base specification defines protocol bindings and an XML container syntax for configuration and management operations, but does not include a data modeling language or accompanying rules for how to model configuration and state information carried by NETCONF. The IETF Operations Area has a long tradition of defining data for SNMP Management Information Bases (MIB) modules [RFC1157] using the Structure of Management Information (SMI) language [RFC2578] to model its data. While this specific modeling approach has a number of well-understood problems, most of the data modeling features provided by SMI are still considered extremely important. Simply modeling the valid syntax without the additional semantic relationships has caused significant interoperability problems in the past.

The NETCONF community concluded that a data modeling framework is needed to support ongoing development of IETF and vendor-defined management information modules. The NETMOD Working Group was chartered to design a modeling language defining the semantics of operational data, configuration data, event notifications and operations, with focus on "human-friendliness", i.e., readability and ease of use. The result is the YANG data modeling language [RFC6020], which now serves for the normative description of NETCONF data models.

Since NETCONF uses XML for encoding its messages, it is natural to express the constraints on NETCONF content using standard XML schema languages. For this purpose, the NETMOD WG selected the Document Schema Definition Languages (DSDL) that is being standardized as ISO/IEC 19757 [DSDL]. The DSDL framework comprises a set of XML schema languages that address grammar rules, semantic constraints and other data modeling aspects, but also, and more importantly, do it in a coordinated and consistent way. While it is true that some DSDL parts have not been standardized yet and are still work in progress, the three parts that the YANG-to-DSDL mapping relies upon - Regular Language for XML Next Generation (RELAX NG), Schematron and Document Schema Renaming Language (DSRL) - already have the status of an ISO/IEC International Standard and are supported in a number of software tools.

This document contains a specification of a mapping that translates YANG data models to XML schemas utilizing a subset of the DSDL schema languages. The mapping procedure is divided into two steps: In the first step, the structure of the data tree, signatures of remote procedure call (RPC) operations and notifications is expressed as the so-called "hybrid schema" - a single RELAX NG schema with annotations

representing additional data model information (metadata, documentation, semantic constraints, default values etc.). The second step then generates a coordinated set of DSDL schemas that can be used for validating specific XML documents such as client requests, server responses or notifications, perhaps also taking into account additional context such as active capabilities or features.

## 2. Terminology and Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The following terms are defined in [RFC4741]:

- o client
- o datastore
- o message
- o operation
- o server

The following terms are defined in [RFC6020]:

- o augment
- o base type
- o built-in type
- o configuration data
- o container
- o data model
- o data node
- o data tree
- o derived type
- o device deviation
- o extension
- o feature
- o grouping
- o instance identifier

- o leaf-list
- o list
- o mandatory node
- o module
- o RPC
- o RPC operation
- o schema node
- o schema tree
- o state data
- o submodule
- o top-level data node
- o uses

The following terms are defined in [XML-INFOSET]:

- o attribute
- o document
- o document element
- o document type declaration (DTD)
- o element
- o information set
- o namespace

In the text, the following typographic conventions are used:

- o YANG statement keywords are delimited by single quotes.
- o XML element names are delimited by "<" and ">" characters.
- o Names of XML attributes are prefixed by the "@" character.

- o Other literal values are delimited by double quotes.

XML elements names are always written with explicit namespace prefixes corresponding to the following XML vocabularies:

"a" DTD compatibility annotations [RNG-DTD];

"dc" Dublin Core metadata elements [RFC5013];

"dsrl" Document Semantics Renaming Language [DSRL];

"en" NETCONF event notifications [RFC5277];

"nc" NETCONF protocol [RFC4741];

"nma" NETMOD-specific schema annotations (see Section 5.3);

"nmf" NETMOD-specific XPath extension functions (see Section 12.7);

"rng" RELAX NG [RNG];

"sch" ISO Schematron [Schematron];

"xsd" W3C XML Schema [XSD].

The following table shows the mapping of these prefixes to namespace URIs.

Prefix	Namespace URI
a	http://relaxng.org/ns/compatibility/annotations/1.0
dc	http://purl.org/dc/terms
dsrl	http://purl.oclc.org/dsdl/dsrl
en	urn:ietf:params:xml:ns:netconf:notification:1.0
nc	urn:ietf:params:xml:ns:netconf:base:1.0
nma	urn:ietf:params:xml:ns:netmod:dSDL-annotations:1
nmf	urn:ietf:params:xml:ns:netmod:xpath-extensions:1
rng	http://relaxng.org/ns/structure/1.0
sch	http://purl.oclc.org/dsdl/schematron
xsd	http://www.w3.org/2001/XMLSchema

Table 1: Used namespace prefixes and corresponding URIs

### 2.1. Glossary of New Terms

- o ancestor datatype: Any datatype a given datatype is (transitively) derived from.
- o ancestor built-in datatype: The built-in datatype that is at the start of the type derivation chain for a given datatype.
- o hybrid schema: A RELAX NG schema with annotations, which embodies the same information as the source YANG module(s). See Section 8.1 for details.
- o implicit node: A data node that, if it is not instantiated in a data tree, may be added to the information set of that data tree (configuration, RPC input or output, notification) without changing the semantics of the data tree.

### 3. Objectives and Motivation

The main objective of this work is to complement YANG as a data modeling language with validation capabilities of DSDL schema languages, namely RELAX NG, Schematron and DSRL. This document describes the correspondence between grammatical, semantic and data type constraints expressed in YANG and equivalent DSDL patterns and rules. The ultimate goal is to be able to capture all substantial information contained in YANG modules and express it in DSDL schemas. While the mapping from YANG to DSDL described in this document may in principle be invertible, the inverse mapping from DSDL to YANG is beyond the scope of this document.

XML-based information models and XML-encoded data appear in several different forms in various phases of YANG data modeling and NETCONF workflow - configuration datastore contents, RPC requests and replies, and notifications. Moreover, RPC operations are characterized by an inherent diversity resulting from selective availability of capabilities and features. YANG modules can also define new RPC operations. The mapping should be able to accommodate this variability and generate schemas that are specifically tailored to a particular situation and thus considerably more effective for validation than generic all-encompassing schemas.

In order to cope with this variability, we assume that the DSDL schemas will be generated on demand for a particular purpose from the available collection of YANG modules and their lifetime will be relatively short. In other words, we don't envision that any collection of DSDL schemas will be created and maintained over an extended period of time in parallel to YANG modules.

The generated schemas are primarily intended as input to existing XML schema validators and other off-the-shelf tools. However, the schemas may also be perused by developers and users as a formal representation of constraints on a particular XML-encoded data object. Consequently, our secondary goal is to keep the schemas as readable as possible. To this end, the complexity of the mapping is distributed into two steps:

1. The first step maps one or more YANG modules to the so-called hybrid schema, which is a single RELAX NG schema that describes grammatical constraints for the main data tree as well as for RPC operations and notifications. Semantic constraints and other information appearing in the input YANG modules is recorded in the hybrid schema in the form of foreign namespace annotations. The output of the first step can thus be considered a virtually complete equivalent of the input YANG modules.

2. In the second step, the hybrid schema from step 1 is transformed further to a coordinated set of fully conformant DSDL schemas containing constraints for a particular data object and a specific situation. The DSDL schemas are intended mainly for machine validation using off-the-shelf tools.



#### 4. DSDL Schema Languages

Document Schema Definition Languages (DSDL) is a framework of schema languages that is being developed as the International Standard ISO/IEC 19757 [DSDL]. Unlike other approaches to XML document validation, most notably W3C XML Schema Definition (XSD) [XSD], the DSDL framework adheres to the principle of "small languages": Each of the DSDL constituents is a stand-alone schema language with a relatively narrow purpose and focus. Together, these schema languages may be used in a coordinated way to accomplish various validation tasks.

The mapping described in this document uses three of the DSDL schema languages, namely RELAX NG [RNG], Schematron [Schematron] and DSRL [DSRL].

##### 4.1. RELAX NG

RELAX NG (pronounced "relaxing") is an XML schema language for grammar-based validation and Part 2 of the ISO/IEC DSDL family of standards [RNG]. Like the W3C XML Schema language [XSD], it is able to describe constraints on the structure and contents of XML documents. However, unlike the DTD [XML] and XSD schema languages, RELAX NG intentionally avoids any infost augmentation such as defining default values. In the DSDL architecture, the particular task of defining and applying default values is delegated to another schema language, DSRL (see Section 4.3).

As its base datatype library, RELAX NG uses the W3C XML Schema Datatype Library [XSD-D], but unlike XSD, other datatype libraries may be used along with it or even replace it if necessary.

RELAX NG is very liberal in accepting annotations from other namespaces. With a few exceptions, such annotations may be placed anywhere in the schema and need no encapsulating elements such as `<xsd:annotation>` in XSD.

RELAX NG schemas can be represented in two equivalent syntaxes: XML and compact. The compact syntax is described in Annex C of the RELAX NG specification [RNG-CS], which was added to the standard in 2006 (Amendment 1). Automatic bidirectional conversions between the two syntaxes can be accomplished using several tools, for example Trang [Trang].

For its terseness and readability, the compact syntax is often the preferred form for publishing RELAX NG schemas whereas validators and other software tools usually work with the XML syntax. However, the compact syntax has two drawbacks:

- o External annotations make the compact syntax schema considerably less readable. While in the XML syntax the annotating elements and attributes are represented in a simple and uniform way (XML elements and attributes from foreign namespaces), the compact syntax uses as many as four different syntactic constructs: documentation, grammar, initial and following annotations. Therefore, the impact of annotations on readability is often much stronger for the compact syntax than it is for the XML syntax.
- o In a computer program, it is more difficult to generate the compact syntax than the XML syntax. While a number of software libraries exist that make it easy to create an XML tree in the memory and then serialize it, no such aid is available for the compact syntax.

For these reasons, the mapping specification in this document uses exclusively the XML syntax. Where appropriate, though, the schemas resulting from the translation MAY be presented in the equivalent compact syntax.

RELAX NG elements are qualified with the namespace URI "http://relaxng.org/ns/structure/1.0". The namespace of the W3C Schema Datatype Library is "http://www.w3.org/2001/XMLSchema-datatypes".

#### 4.2. Schematron

Schematron is Part 3 of DSDL that reached the status of a full ISO/IEC standard in 2006 [Schematron]. In contrast to the traditional schema languages such as DTD, XSD or RELAX NG, which are based on the concept of a formal grammar, Schematron utilizes a rule-based approach. Its rules may specify arbitrary conditions involving data from different parts of an XML document. Each rule consists of three essential components:

- o context - an XPath expression that defines the set of locations where the rule is to be applied;
- o assert or report condition - another XPath expression that is evaluated relative to the location matched by the context expression;
- o human-readable message that is displayed when the assert condition is false or report condition is true.

The difference between the assert and report condition is that the former is positive in that it states a condition that a valid document has to satisfy, whereas the latter specifies an error

condition.

Schematron draws most of its expressive power from XPath [XPath] and Extensible Stylesheet Language Transformations (XSLT) [XSLT]. ISO Schematron allows for dynamic query language binding so that the following XML query languages can be used: STX, XSLT 1.0, XSLT 1.1, EXSLT, XSLT 2.0, XPath 1.0, XPath 2.0 and XQuery 1.0 (this list may be extended in the future).

Human-readable error messages are another feature that sets Schematron apart from other common schema languages. The messages may even contain XPath expressions that are evaluated in the actual context and thus refer to information items in the XML document being validated.

Another feature of Schematron that is used by the mapping are abstract patterns. These work essentially as macros and may also contain parameters which are supplied when the abstract pattern is used.

Schematron elements are qualified with namespace URI "http://purl.oclc.org/dsdl/schematron".

#### 4.3. Document Semantics Renaming Language (DSRL)

DSRL (pronounced "disrule") is Part 8 of DSDL that reached the status of a full ISO/IEC standard in 2008 [DSRL]. Unlike RELAX NG and Schematron, DSRL is allowed to modify XML information set of the validated document. While DSRL is primarily intended for renaming XML elements and attributes, it can also define default values for XML attributes and default contents for XML elements or subtrees so that the default contents are inserted if they are missing in the validated documents. The latter feature is used by the YANG-to-DSDL mapping for representing YANG default contents consisting of leaf nodes with default values and their ancestor non-presence containers.

DSRL elements are qualified with namespace URI "http://purl.oclc.org/dsdl/dsrl".

## 5. Additional Annotations

Besides the DSDL schema languages, the mapping also uses three sets of annotations that are added as foreign-namespace attributes and elements to RELAX NG schemas.

Two of the annotation sets - Dublin Core elements and DTD compatibility annotations - are standard vocabularies for representing metadata and documentation, respectively. Although these data model items are not used for formal validation, they quite often carry important information for data model implementers. Therefore, they SHOULD be included in the hybrid schema and MAY also appear in the final validation schemas.

The third set are NETMOD-specific annotations. They are specifically designed for the hybrid schema and convey semantic constraints and other information that cannot be expressed directly in RELAX NG. In the second mapping step, these annotations are converted to Schematron and DSRL rules.

### 5.1. Dublin Core Metadata Elements

Dublin Core is a system of metadata elements that was originally created for describing metadata of World Wide Web resources in order to facilitate their automated lookup. Later it was accepted as a standard for describing metadata of arbitrary resources. This specification uses the definition from [RFC5013].

Dublin Core elements are qualified with namespace URI "http://purl.org/dc/terms".

### 5.2. RELAX NG DTD Compatibility Annotations

DTD compatibility annotations are a part of the RELAX NG DTD Compatibility specification [RNG-DTD]. YANG-to-DSDL mapping uses only the <a:documentation> annotation for representing YANG 'description' and 'reference' texts.

Note that there is no intention to make the resulting schemas DTD-compatible, the main reason for using these annotations is technical: they are well supported and adequately formatted by several RELAX NG tools.

DTD compatibility annotations are qualified with namespace URI "http://relaxng.org/ns/compatibility/annotations/1.0".

### 5.3. NETMOD-Specific Annotations

NETMOD-specific annotations are XML elements and attributes qualified with the namespace URI "urn:ietf:params:xml:ns:netmod:dSDL-annotations:1" which appear in various locations of the hybrid schema. YANG statements are mapped to these annotations in a straightforward way. In most cases, the annotation attributes and elements have the same name as the corresponding YANG statement.

Table 2 lists alphabetically the names of NETMOD-specific annotation attributes (prefixed with "@") and elements (in angle brackets) along with a reference to the section where their use is described. Appendix A contains a RELAX NG schema for this annotation vocabulary.

annotation	section	note
@nma:config	10.9	
<nma:data>	8.1	4
@nma:default	10.12	
<nma:error-app-tag>	10.16	1
<nma:error-message>	10.17	1
@nma:if-feature	10.22	
@nma:implicit	10.11, 10.7, 10.12	
<nma:input>	8.1	4
<nma:instance-identifier>	10.53.7	2
@nma:key	10.26	
@nma:leaf-list	10.28	
@nma:leafref	10.53.8	
@nma:mandatory	10.8	
@nma:max-elements	10.28	
@nma:min-elements	10.28	

@nma:module	10.34	
<nma:must>	10.35	3
<nma:notification>	8.1	4
<nma:notifications>	8.1	4
@nma:ordered-by	10.38	
<nma:output>	8.1	4
<nma:rpc>	8.1	4
<nma:rpcs>	8.1	4
@nma:status	10.51	
@nma:unique	10.55	
@nma:units	10.56	
@nma:when	10.59	

Table 2: NETMOD-specific annotations

## Notes:

1. Appears only as a subelement of <nma:must>.
2. Has an optional attribute @require-instance.
3. Has a mandatory attribute @assert and two optional subelements <nma:error-app-tag> and <nma:error-message>.
4. Marker element in the hybrid schema.

## 6. Overview of the Mapping

This section gives an overview of the YANG-to-DSDL mapping, its inputs and outputs. Figure 1 presents an overall structure of the mapping:

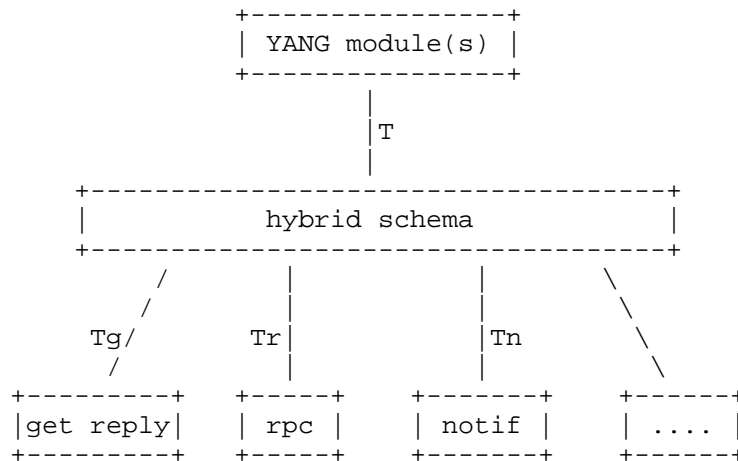


Figure 1: Structure of the mapping

The mapping procedure is divided into two steps:

1. Transformation T in the first step maps one or more YANG modules to the hybrid schema (see Section 8.1). Constraints that cannot be expressed directly in RELAX NG (list key definitions, 'must' statements etc.) and various documentation texts are recorded in the schema as foreign-namespace annotations.
2. In the second step, the hybrid schema may be transformed in multiple ways to a coordinated set of DSDL schemas that can be used for validating a particular data object in a specific context. Figure 1 shows three simple possibilities as examples. In the process, appropriate parts of the hybrid schema are extracted and specific annotations transformed to equivalent, but usually more complex, Schematron patterns, DSRL element maps etc.

An implementation of the mapping algorithm MUST accept one or more valid YANG modules as its input. It is important to be able to process multiple YANG modules together since multiple modules may be negotiated for a NETCONF session and the contents of the configuration datastore is then obtained as the union of data trees specified by the individual modules, which may also lead to multiple root nodes of the datastore hierarchy. In addition, the input

modules may be further coupled by the 'augment' statement in which one module augments the data tree of another module.

It is also assumed that the algorithm has access, perhaps on demand, to all YANG modules that the input modules import (directly or transitively).

Other information contained in input YANG modules, such as semantic constraints and default values, are recorded in the hybrid schema as annotations - XML attributes or elements qualified with namespace URI "urn:ietf:params:xml:ns:netmod:dSDL-annotations:1". Metadata describing the YANG modules are mapped to Dublin Core annotations elements (Section 5.1). Finally, documentation strings are mapped to <a:documentation> elements belonging to the DTD compatibility vocabulary (Section 5.2).

The output of the second step is a coordinated set of three DSDL schemas corresponding to a specific data object and context:

- o RELAX NG schema describing the grammatical and datatype constraints;
- o Schematron schema expressing other constraints such as uniqueness of list keys or user-specified semantic rules;
- o DSRL schema containing the specification of default contents.



## 7. NETCONF Content Validation

This section describes how the schemas generated by the YANG-to-DSDL mapping are supposed to be applied for validating XML instance documents such as the contents of a datastore or various NETCONF messages.

The validation proceeds in the following steps, which are also illustrated in Figure 2:

1. The XML instance document is checked for grammatical and data type validity using the RELAX NG schema.
2. Default values for leaf nodes have to be applied and their ancestor containers added where necessary. It is important to add the implicit nodes before the next validation step because YANG specification [RFC6020] requires that the data tree against which XPath expressions are evaluated already has all defaults filled-in. Note that this step modifies the information set of the validated XML document.
3. The semantic constraints are checked using the Schematron schema.

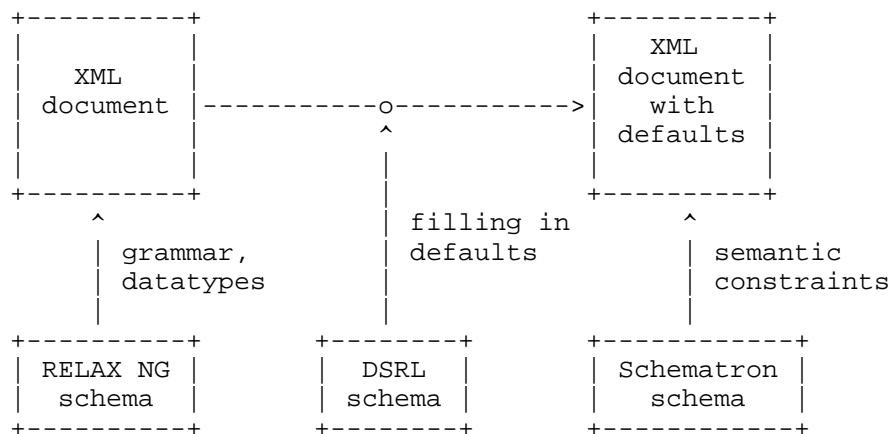


Figure 2: Outline of the validation procedure

## 8. Design Considerations

YANG data models could in principle be mapped to the DSDL schemas in a number of ways. The mapping procedure described in this document uses several specific design decisions that are discussed in the following subsections.

### 8.1. Hybrid Schema

As was explained in Section 6, the first step of the mapping produces an intermediate document - the hybrid schema, which specifies all constraints for the entire data model in a single RELAX NG schema.

Every input YANG module corresponds to exactly one embedded grammar in the hybrid schema. This separation of input YANG modules allows each embedded grammar to include named pattern definitions into its own namespace, which is important for mapping YANG groupings (see Section 9.2 for additional details).

In addition to grammatical and datatype constraints, YANG modules provide other important information that cannot be expressed in a RELAX NG schema: semantic constraints, default values, metadata, documentation and so on. Such information items are represented in the hybrid schema as XML attributes and elements belonging to the namespace with the following URI:  
"urn:ietf:params:xml:ns:netmod:dSDL-annotations:1". A complete list of these annotations is given in Section 5.3, detailed rules about their use are then contained in the following sections.

YANG modules define data models not only for configuration and state data but also for (multiple) RPC operations [RFC4741] and/or event notifications [RFC5277]. In order to be able to capture all three types of data models in one schema document, the hybrid schema uses special markers that enclose sub-schemas for configuration and state data, individual RPC operations (both input and output part) and individual notifications.

The markers are the following XML elements in the namespace of NETMOD-specific annotations (URI  
urn:ietf:params:xml:ns:netmod:dSDL-annotations:1):

Element name	Role
nma:data	encloses configuration and state data
nma:rpcs	encloses all RPC operations
nma:rpc	encloses an individual RPC operation
nma:input	encloses an RPC request
nma:output	encloses an RPC reply
nma:notifications	encloses all notifications
nma:notification	encloses an individual notification

Table 3: Marker elements in the hybrid schema

For example, consider a data model formed by two YANG modules "example-a" and "example-b" that define nodes in the namespaces "http://example.com/ns/example-a" and "http://example.com/ns/example-b". Module "example-a" defines configuration/state data, RPC methods and notifications, whereas "example-b" defines only configuration/state data. The hybrid schema can then be schematically represented as follows:

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:nma="urn:ietf:params:xml:ns:netmod:dSDL-annotations:1"
  xmlns:exa="http://example.com/ns/example-a"
  xmlns:exb="http://example.com/ns/example-b"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <start>
    <grammar nma:module="example-a"
      ns="http://example.com/ns/example-a">
      <start>
        <nma:data>
          ...configuration and state data defined in "example-a"...
        </nma:data>
        <nma:rpcs>
          <nma:rpc>
            <nma:input>
              <element name="exa:myrpc">
                ...
              </element>
            </nma:input>
            <nma:output>
```

```

        ...
        </nma:output>
    </nma:rpc>
    ...
</nma:rpcs>
<nma:notifications>
    <nma:notification>
        <element name="exa:mynotif">
            ...
            </element>
        </nma:notification>
    ...
</nma:notifications>
</start>
...local named pattern definitions of example-a...
</grammar>
<grammar nma:module="example-b"
    ns="http://example.com/ns/example-a">
    <start>
        <nma:data>
            ...configuration and state data defined in "example-b"...
        </nma:data>
        <nma:rpcs/>
        <nma:notifications/>
    </start>
    ...local named pattern definitions of example-b...
</grammar>
</start>
...global named pattern definitions...
</grammar>

```

A complete hybrid schema for the data model of a DHCP server is given in Appendix C.2.

## 8.2. Modularity

Both YANG and RELAX NG offer means for modularity, i.e., for splitting the contents of a full schema into separate modules and combining or reusing them in various ways. However, the approaches taken by YANG and RELAX NG differ. Modularity in RELAX NG is suitable for ad hoc combinations of a small number of schemas whereas YANG assumes a large set of modules similar to SNMP MIB modules. The following differences are important:

- o In YANG, whenever module A imports module B, it gets access to the definitions (groupings and typedefs) appearing at the top level of module B. However, no part of data tree from module B is imported along with it. In contrast, the `<rng:include>` pattern in RELAX NG

imports both definitions of named patterns and the entire schema tree from the included schema.

- o The names of imported YANG groupings and typedefs are qualified with the namespace of the imported module. On the other hand, the names of data nodes contained inside the imported groupings, when used within the importing module, become part of the importing module's namespace. In RELAX NG, the names of patterns are unqualified and so named patterns defined in both the importing and imported module share the same flat namespace. The contents of RELAX NG named patterns may either keep the namespace of the schema where they are defined or inherit the namespace of the importing module, analogically to YANG. However, in order to achieve the latter behavior, the definitions of named patterns must be included from an external schema which has to be prepared in a special way (see [Vli04], Chapter 11).

In order to map, as much as possible, the modularity of YANG to RELAX NG, a validating RELAX NG schema (the result of the second mapping step) has to be split into two files, one of them containing all global definitions that are mapped from top-level YANG groupings appearing in all input YANG module. This RELAX NG schema MUST NOT define any namespace via the @ns attribute.

The other RELAX NG schema file then defines actual data trees mapped from input YANG modules, each of them enclosed in an own embedded grammar. Those embedded grammars in which at least one of the global definitions is used MUST include the first schema with definitions and also MUST define the local namespace using the @ns attribute. This way, the global definitions can be used inside different embedded grammar, each time accepting a different local namespace.

Named pattern definition that are mapped from non-top-level YANG groupings MUST be placed inside the embedded grammar corresponding to the YANG module where the grouping is defined.

In the hybrid schema, we need to distinguish the global and non-global named pattern definitions while still keeping the hybrid schema in one file. This is accomplished in the following way:

- o Every global definition MUST be placed as a child of the the outer <rng:grammar> element (the document root of the hybrid schema).
- o Every non-global definitions MUST be placed as a child of the corresponding embedded <rng:grammar> element.

YANG also allows for splitting a module into a number of submodules. However, as submodules have no impact on the scope of identifiers and

namespaces, the modularity based on submodules is not mapped in any way. The contents of submodules is therefore handled as if the submodule text appeared directly in the main module.

### 8.3. Granularity

RELAX NG supports different styles of schema structuring: One extreme, often called "Russian Doll", specifies the structure of an XML instance document in a single hierarchy. The other extreme, the flat style, uses a similar approach as the Data Type Definition (DTD) schema language - every XML element corresponds to a named pattern definition. In practice, some compromise between the two extremes is usually chosen.

YANG supports both styles in principle, too, but in most cases the modules are organized in a way closer to the "Russian Doll" style, which provides a better insight into the structure of the configuration data. Groupings are usually defined only for contents that are prepared for reuse in multiple places via the 'uses' statement. In contrast, RELAX NG schemas tend to be much flatter, because finer granularity is also needed in RELAX NG for extensibility of the schemas - it is only possible to replace or modify schema fragments that are factored out as named patterns. For YANG this is not an issue since its 'augment' and 'refine' statements can delve, by using path expressions, into arbitrary depths of existing structures.

In general, it is not feasible to map YANG's powerful extension mechanisms to those available in RELAX NG. For this reason, the mapping essentially keeps the granularity of the original YANG data model: YANG groupings and definitions of derived types usually have direct counterparts in definitions of named patterns in the resulting RELAX NG schema.

### 8.4. Handling of XML Namespaces

Most modern XML schema languages, including RELAX NG, Schematron and DSRL, support schemas for so-called compound XML documents which contain elements from multiple namespaces. This is useful for our purpose since the YANG-to-DSDL mapping allows for multiple input YANG modules, which naturally leads to compound document schemas.

RELAX NG offers two alternatives for defining the target namespaces in the schema:

1. First possibility is the traditional XML way via the @xmlns:xxx attribute.

2. One of the target namespace URIs may be declared using the @ns attribute.

In both the hybrid schema and validation RELAX NG schemas generated in the second step, the namespaces MUST be declared as follows:

1. The root <rng:grammar> MUST have @xmlns:xxx attributes declaring prefixes of all namespaces that are used in the data model. The prefixes SHOULD be identical to those defined in the 'prefix' statements. An implementation of the mapping MUST resolve all collisions in the prefixes defined by different input modules, if there are any.
2. Each embedded <rng:grammar> element MUST declare the namespace of the corresponding module using the @ns attribute. This way, the names of nodes defined by global named patterns are able to adopt the local namespace of each embedded grammar, as explained in Section 8.2.

This setup is illustrated by the example at the end of Section 8.1.

DSRL schemas may declare any number of target namespaces via the standard XML attributes xmlns:xxx.

In contrast, Schematron requires all used namespaces to be defined in the <sch:ns> subelements of the document element <sch:schema>.

## 9. Mapping YANG Data Models to the Hybrid Schema

This section explains the main principles governing the first step of the mapping. Its result is the hybrid schema which is described in Section 8.1.

A detailed specification of the mapping of individual YANG statements is contained in the following Section 10.

### 9.1. Occurrence Rules for Data Nodes

In DSDL schema languages, occurrence constraints for a node are always localized together with that node. In a RELAX NG schema, for example, `<rng:optional>` pattern appears as the parent element of the pattern defining a leaf or non-leaf element. Similarly, DSRL specifies default contents separately for every single node, be it a leaf or non-leaf element.

For leaf nodes in YANG modules, the occurrence constraints are also easily inferred from the substatements of 'leaf'. On the other hand, for a YANG container it is often necessary to examine its entire subtree in order to determine the container's occurrence constraints.

Therefore, one of the goals of the first mapping step is to infer the occurrence constraints for all data nodes and mark accordingly the corresponding `<rng:element>` patterns in the hybrid schema so that any transformation procedure in the second mapping step can simply use this information and need not examine the subtree again.

First, it has to be decided whether a given data node must always be present in a valid configuration. If so, such a node is called mandatory, otherwise it is called optional. This constraint is closely related to the notion of mandatory nodes in Section 3.1 in [RFC6020]. The only difference is that this document also considers list keys to be mandatory.

The other occurrence constraint has to do with the semantics of the 'default' statement and the possibility of removing empty non-presence containers. As a result, the information set of a valid configuration may be modified by adding or removing certain leaf or container elements without changing the meaning of the configuration. In this document, such elements are called implicit. In the hybrid schema, they can be identified as RELAX NG patterns having either `@nma:default` or `@nma:implicit` attribute.

Note that both occurrence constraints apply to containers at the top level of the data tree, and then also to other containers under the additional condition that their parent node exists in the instance



document. For example, consider the following YANG fragment:

```
container outer {
  presence 'Presence of "outer" means something.';
  container c1 {
    leaf foo {
      type uint8;
      default 1;
    }
  }
  container c2 {
    leaf-list bar {
      type uint8;
      min-elements 0;
    }
  }
  container c3 {
    leaf baz {
      type uint8;
      mandatory true;
    }
  }
}
```

Here, container "outer" has the 'presence' substatement, which means that it is optional and not implicit. If "outer" is not present in a configuration, its child containers are not present as well. However, if "outer" does exist, it makes sense to ask which of its child containers are optional and which are implicit. In this case, "c1" is optional and implicit, "c2" is optional but not implicit and "c3" is mandatory (and therefore not implicit).

The following subsections give precise rules for determining whether a container is optional or mandatory and whether it is implicit. In order to simplify the recursive definition of these occurrence characteristics, it is useful to define them also for other types of YANG schema nodes, i.e., leaf, list, leaf-list and anyxml and choice.

#### 9.1.1.1. Optional and Mandatory Nodes

The decision whether a given node is mandatory or optional is governed by the following rules:

- o Leaf, anyxml and choice nodes are mandatory if they contain the substatement "mandatory true;". For a choice node this means that at least one node from exactly one case branch must exist.

- o In addition, a leaf node is mandatory if it is declared as a list key.
- o A list or leaf-list node is mandatory if it contains the 'min-elements' substatement with an argument value greater than zero.
- o A container node is mandatory if its definition does not contain the 'presence' substatement and at least one of its child nodes is mandatory.

A node which is not mandatory is said to be optional.

In RELAX NG, definitions of nodes that are optional must be explicitly wrapped in the `<rng:optional>` element. The mapping MUST use the above rules to determine whether a YANG node is optional and if so, insert the `<rng:optional>` element in the hybrid schema.

However, alternatives in `<rng:choice>` MUST NOT be defined as optional in the hybrid schema. If a choice in YANG is not mandatory, `<rng:optional>` MUST be used to wrap the entire `<rng:choice>` pattern.

#### 9.1.2. Implicit Nodes

The following rules are used to determine whether a given data node is implicit:

- o List, leaf-list and anyxml nodes are never implicit.
- o A leaf node is implicit if and only if it has a default value, defined either directly or via its datatype.
- o A container node is implicit if and only if it does not have the 'presence' substatement, none of its children are mandatory and at least one child is implicit.

In the hybrid schema, all implicit containers, as well as leafs that obtain their default value from a typedef and don't have the `@nma:default` attribute, MUST be marked with `@nma:implicit` attribute having the value of "true".

Note that Section 7.9.3 in [RFC6020] specifies other rules that must be taken into account when deciding whether a given container or leaf appearing inside a case of a choice is ultimately implicit or not. Specifically, a leaf or container under a case can be implicit only if the case appears in the argument of the choice's 'default' statement. However, this is not sufficient by itself but also depends on the particular instance XML document, namely on the presence or absence of nodes from other (non-default) cases. The

details are explained in Section 11.3.

## 9.2. Mapping YANG Groupings and Typedefs

YANG groupings and typedefs are generally mapped to RELAX NG named patterns. There are, however, several caveats that the mapping has to take into account.

First of all, YANG typedefs and groupings may appear at all levels of the module hierarchy and are subject to lexical scoping, see Section 5.5 in [RFC6020]. Second, top-level symbols from external modules may be imported as qualified names represented using the external module namespace prefix and the name of the symbol. In contrast, named patterns in RELAX NG (both local and imported via the `<rng:include>` pattern) share the same namespace and within a grammar they are always global - their definitions may only appear at the top level as children of the `<rng:grammar>` element. Consequently, whenever YANG groupings and typedefs are mapped to RELAX NG named pattern definitions, their names MUST be disambiguated in order to avoid naming conflicts. The mapping uses the following procedure for mangling the names of groupings and type definitions:

- o Names of groupings and typedefs appearing at the top level of the YANG module hierarchy are prefixed with the module name and two underscore characters ("`__`").
- o Names of other groupings and typedefs, i.e., those that do not appear at the top level of a YANG module, are prefixed with the module name, double underscore, and then the names of all ancestor data nodes separated by double underscore.
- o Finally, since the names of groupings and typedefs in YANG have different namespaces, an additional underscore character is added to the beginning of the mangled names of all groupings.

An additional complication is caused by the YANG rules for subelement ordering (see, e.g., Section 7.5.7 in [RFC6020]): In RPC input and output parameters, subelements must follow the order specified in the data model, otherwise the order is arbitrary. Consequently, if a grouping is used both in RPC input/output parameters and elsewhere, it MUST be mapped to two different named pattern definitions - one with fixed order and the other with arbitrary order. To distinguish them, the "`__rpc`" suffix MUST be appended to the version with fixed order.

EXAMPLE. Consider the following YANG module which imports the standard module "ietf-inet-types" [RFC6021]:

```
module example1 {
  namespace "http://example.com/ns/example1";
  prefix ex1;
  typedef vowels {
    type string {
      pattern "[aeiouy]*";
    }
  }
  grouping "grp1" {
    leaf "void" {
      type "empty";
    }
  }
  container "cont" {
    leaf foo {
      type vowels;
    }
    uses "grp1";
  }
}
```

The hybrid schema generated by the first mapping step will then contain the following two (global) named pattern definitions:

```
<rng:define name="example1__vowels">
  <rng:data type="string">
    <rng:param name="pattern">[aeiouy]*</rng:param>
  </rng:data>
</rng:define>

<rng:define name="_example1__grp1">
  <rng:optional>
    <rng:element name="void">
      <rng:empty/>
    </rng:element>
  </rng:optional>
</rng:define>
```

#### 9.2.1. YANG Refinements and Augments

YANG groupings represent a similar concept as named pattern definitions in RELAX NG and both languages also offer mechanisms for their subsequent modification. However, in RELAX NG the definitions themselves are modified whereas YANG provides two substatements of 'uses' which modify expansions of groupings:

- o 'refine' statement allows for changing parameters of a schema node inside the grouping referenced by the parent 'uses' statement;

- o 'augment' statement can be used for adding new schema nodes to the grouping contents.

Both 'refine' and 'augment' statements are quite powerful in that they can address, using XPath-like expressions as their arguments, schema nodes that are arbitrarily deep inside the grouping contents. In contrast, modifications of named pattern definitions in RELAX NG are applied exclusively at the topmost level of the named pattern contents. In order to achieve a modifiability of named patterns comparable to YANG, a RELAX NG schema would have to be extremely flat (cf. Section 8.3) and very difficult to read.

Since the goal of the mapping described in this document is to generate ad hoc DSDL schemas, we decided to avoid these complications and instead expand the grouping and refine and/or augment it "in place". In other words, every 'uses' statement which has 'refine' and/or 'augment' substatements is replaced by the contents of the corresponding grouping, the changes specified in the 'refine' and 'augment' statements are applied and the resulting YANG schema fragment is mapped as if the 'uses'/'grouping' indirection wasn't there.

If there are further 'uses' statements inside the grouping contents, they may require expansion, too: it is necessary if the contained 'uses'/'grouping' pair lies on the "modification path" specified in the argument of a 'refine' or 'augment' statement.

EXAMPLE. Consider the following YANG module:

```
module example2 {
  namespace "http://example.com/ns/example2";
  prefix ex2;
  grouping leaves {
    uses fr;
    uses es;
  }
  grouping fr {
    leaf feuille {
      type string;
    }
  }
  grouping es {
    leaf hoja {
      type string;
    }
  }
  uses leaves;
}
```

The resulting hybrid schema contains three global named pattern definitions corresponding to the three groupings, namely

```
<rng:define name="_example2__leaves">
  <rng:interleave>
    <rng:ref name="_example2__fr"/>
    <rng:ref name="_example2__es"/>
  </rng:interleave>
</rng:define>
```

```
<rng:define name="_example2__fr">
  <rng:optional>
    <rng:element name="feuille">
      <rng:data type="string"/>
    </rng:element>
  </rng:optional>
</rng:define>
```

```
<rng:define name="_example2__es">
  <rng:optional>
    <rng:element name="hoja">
      <rng:data type="string"/>
    </rng:element>
  </rng:optional>
</rng:define>
```

and the configuration data part of the hybrid schema is a single named pattern reference:

```
<nma:data>
  <rng:ref name="_example2__leaves"/>
</nma:data>
```

Now assume that the "uses leaves" statement contains a 'refine' substatement, for example:

```
uses leaves {
  refine "hoja" {
    default "alamo";
  }
}
```

The resulting hybrid schema now contains just one named pattern definition - "\_example2\_\_fr". The other two groupings "leaves" and "es" have to be expanded because they both lie on the "modification path", i.e., contain the leaf "hoja" that is being refined. The configuration data part of the hybrid schema now looks like this:

```
<nma:data>
  <rng:interleave>
    <rng:ref name="_example2__fr"/>
    <rng:optional>
      <rng:element name="ex2:hoja" nma:default="alamo">
        <rng:data type="string"/>
      </rng:element>
    </rng:optional>
  </rng:interleave>
</nma:data>
```

#### 9.2.2. Type Derivation Chains

RELAX NG has no equivalent of the type derivation mechanism in YANG that allows to restrict a built-in type (perhaps in multiple steps) by adding new constraints. Whenever a derived YANG type is used without restrictions - as a substatement of either 'leaf' or another 'typedef' - then the 'type' statement is mapped simply to a named pattern reference <rng:ref>, and the type definition is mapped to a RELAX NG named pattern definition <rng:define>. However, if any restrictions are specified as substatements of the 'type' statement, the type definition MUST be expanded at that point so that only the ancestor built-in type appears in the hybrid schema, restricted with facets that correspond to the combination of all restrictions found along the type derivation chain and also in the 'type' statement.

EXAMPLE. Consider this YANG module:

```
module example3 {
  namespace "http://example.com/ns/example3";
  prefix ex3;
  typedef dozen {
    type uint8 {
      range 1..12;
    }
  }
  leaf month {
    type dozen;
  }
}
```

The 'type' statement in "leaf month" has no restrictions and is therefore mapped simply to the reference <rng:ref name="example3\_\_dozen"/> and the corresponding named pattern is defined as follows:

```
<rng:define name="example3__dozen">
  <rng:data type="unsignedByte">
    <rng:param name="minInclusive">1</rng:param>
    <rng:param name="maxInclusive">12</rng:param>
  </rng:data>
</rng:define>
```

Assume now that the definition of leaf "month" is changed to

```
leaf month {
  type dozen {
    range 7..max;
  }
}
```

The output RELAX NG schema then will not contain any named pattern definition and the leaf "month" will be mapped directly to

```
<rng:element name="ex3:month">
  <rng:data type="unsignedByte">
    <rng:param name="minInclusive">7</rng:param>
    <rng:param name="maxInclusive">12</rng:param>
  </rng:data>
</rng:element>
```

The mapping of type derivation chains may be further complicated by the presence of the 'default' statement in type definitions. In the simple case, when a type definition containing the 'default' statement is used without restrictions, the 'default' statement is mapped to the @nma:default attribute attached to the <rng:define> element.

However, if that type definition has to be expanded due to restrictions, the @nma:default annotation arising from the expanded type or ancestor types in the type derivation chain MUST be attached to the pattern where the expansion occurs. If there are multiple 'default' statements in consecutive steps of the type derivation, only the 'default' statement that is closest to the expanded type is used.

EXAMPLE. Consider this variation of the last example:



```
module example3bis {
  namespace "http://example.com/ns/example3bis";
  prefix ex3bis;
  typedef dozen {
    type uint8 {
      range 1..12;
    }
    default 7;
  }
  leaf month {
    type dozen;
  }
}
```

The 'typedef' statement in this module is mapped to the following named pattern definition:

```
<rng:define name="example3bis__dozen" @nma:default="7">
  <rng:data type="unsignedByte">
    <rng:param name="minInclusive">1</rng:param>
    <rng:param name="maxInclusive">12</rng:param>
  </rng:data>
</rng:define>
```

If the "dozen" type is restricted when used in the leaf "month" definition as in the previous example, the "dozen" type has to be expanded and @nma:default becomes an attribute of the <ex3bis:month> element definition:

```
<rng:element name="ex3bis:month" @nma:default="7">
  <rng:data type="unsignedByte">
    <rng:param name="minInclusive">7</rng:param>
    <rng:param name="maxInclusive">12</rng:param>
  </rng:data>
</rng:element>
```

However, if the definition of the leaf "month" itself contained the 'default' substatement, the default specified for the "dozen" type would be ignored.

### 9.3. Translation of XPath Expressions

YANG uses full XPath 1.0 syntax [XPath] for the arguments of 'must', 'when' and 'path' statements. As the names of data nodes defined in a YANG module always belong to the namespace of that YANG module, YANG adopted a simplification similar to the concept of default namespace in XPath 2.0: node names in XPath expressions needn't carry a namespace prefix inside the module where they are defined and the

local module's namespace is assumed.

Consequently, all XPath expressions MUST be translated into a fully conformant XPath 1.0 expression: Every unprefixed node name MUST be prepended with the local module's namespace prefix as declared by the 'prefix' statement.

XPath expressions appearing inside top-level groupings require special attention because all unprefixed node names contained in them must adopt the namespace of each module where the grouping is used (cf. Section 8.2. In order to achieve this, the local prefix MUST be represented using the variable "\$pref" in the hybrid schema. A Schematron schema which encounters such an XPath expression then supplies an appropriate value for this variable via a parameter to an abstract pattern to which the YANG grouping is mapped (see Section 11.2).

For example, XPath expression "/dhcp/max-lease-time" appearing in a YANG module with the "dhcp" prefix will be translated to

- o "\$pref:dhcp/\$pref:max-lease-time", if the expression is inside a top-level grouping;
- o "dhcp:dhcp/dhcp:max-lease-time", otherwise.

YANG also uses other XPath-like expressions, namely key identifiers and "descendant schema node identifiers" (see the ABNF production for and "descendant-schema-nodeid" in Section 12 of [RFC6020]). These expressions MUST be translated by adding local module prefixes as well.

#### 9.4. YANG Language Extensions

YANG allows for extending its own language in-line by adding new statements with keywords from special namespaces. Such extensions first have to be declared using the 'extension' statement and then they can be used as the standard YANG statements, from which they are distinguished by a namespace prefix qualifying the extension keyword. RELAX NG has a similar extension mechanism - XML elements and attributes with names from foreign namespaces may be inserted at almost any place of a RELAX NG schema.

YANG language extensions may or may not have a meaning in the context of DSDL schemas. Therefore, an implementation MAY ignore any or all of the extensions. However, an extension that is not ignored MUST be mapped to XML element(s) and/or attribute(s) that exactly match the YIN form of the extension, see Section 11.1 in [RFC6020].

EXAMPLE. Consider the following extension defined by the "acme" module:

```
extension documentation-flag {  
    argument number;  
}
```

This extension can then be used in the same or another module, for instance like this:

```
leaf folio {  
    acme:documentation-flag 42;  
    type string;  
}
```

If this extension is honored by the mapping, it will be mapped to

```
<rng:element name="acme:folio">  
    <acme:documentation-flag number="42"/>  
    <rng:data type="string"/>  
</rng:element>
```

Note that the 'extension' statement itself is not mapped in any way.

## 10. Mapping YANG Statements to the Hybrid Schema

Each subsection in this section is devoted to one YANG statement and provides the specification of how the statement is mapped to the hybrid schema. The subsections are sorted alphabetically by the statement keyword.

Each YANG statement is mapped to an XML fragment, typically a single element or attribute but it may also be a larger structure. The mapping procedure is inherently recursive, which means that after finishing a statement the mapping continues with its substatements, if there are any, and a certain element of the resulting fragment becomes the parent of other fragments resulting from the mapping of substatements. Any changes to this default recursive procedure are explicitly specified.

YANG XML encoding rules translate to the following rules for ordering multiple subelements:

1. Within the `<nma:rpcs>` subtree (i.e., for input and output parameters of an RPC operation) the order of subelements is fixed and their definitions in the hybrid schema **MUST** follow the order specified in the source YANG module.
2. When mapping the 'list' statement, all keys **MUST** come before any other subelements and in the same order as they are declared in the 'key' statement. The order of the remaining (non-key) subelements is not specified, so their definitions in the hybrid schema **MUST** be enclosed in the `<rng:interleave>` element.
3. Otherwise, the order of subelements is arbitrary and, consequently, all definitions of subelements in the hybrid schema **MUST** be enclosed in the `<rng:interleave>` element.

The following conventions are used in this section:

- o The argument of the statement being mapped is denoted by `ARGUMENT`.
- o The element in the RELAX NG schema that becomes the parent of the resulting XML fragment is denoted by `PARENT`.

### 10.1. The 'anyxml' Statement

This statement is mapped to `<rng:element>` element and `ARGUMENT` with prepended local namespace prefix becomes the value of its `@name` attribute. The contents of `<rng:element>` are

```
<rng:ref name="__anyxml__"/>
```

Substatements of the 'anyxml' statement, if any, MAY be mapped to additional children of the <rng:element> element.

If at least one 'anyxml' statement occurs in any of the input YANG modules, the following pattern definition MUST be added exactly once to the RELAX NG schema as a child of the root <rng:grammar> element (cf. [Vli04], p. 172):

```
<rng:define name="__anyxml__">
  <rng:zeroOrMore>
    <rng:choice>
      <rng:attribute>
        <rng:anyName/>
      </rng:attribute>
      <rng:element>
        <rng:anyName/>
        <rng:ref name="__anyxml__"/>
      </rng:element>
      <rng:text/>
    </rng:choice>
  </rng:zeroOrMore>
</rng:define>
```

EXAMPLE: YANG statement in a module with namespace prefix "yam"

```
anyxml data {
  description "Any XML content allowed here.";
}
```

is mapped to the following fragment:

```
<rng:element name="yam:data">
  <a:documentation>Any XML content allowed here</a:documentation>
  <rng:ref name="__anyxml__"/>
</rng:element>
```

An anyxml node is optional if there is no "mandatory true;" substatement. The <rng:element> element then MUST be wrapped in <rng:optional>, except when the 'anyxml' statement is a child of the 'choice' statement and thus forms a shorthand case for that choice (see Section 9.1.1 for details).

## 10.2. The 'argument' Statement

This statement is not mapped to the output schema, but see the rules for handling extensions in Section 9.4.

### 10.3. The 'augment' Statement

As a substatement of 'uses', this statement is handled as a part of 'uses' mapping, see Section 10.57.

At the top level of a module or submodule, the 'augment' statement is used for augmenting the schema tree of another YANG module. If the augmented module is not processed within the same mapping session, the top-level 'augment' statement MUST be ignored. Otherwise, the contents of the statement are added to the foreign module with the namespace of the module where the 'augment' statement appears.

### 10.4. The 'base' Statement

This statement is ignored as a substatement of 'identity' and handled within the 'identityref' type if it appears as a substatement of that type definition, see Section 10.53.6.

### 10.5. The 'belongs-to' Statement

This statement is not used since the processing of submodules is always initiated from the main module, see Section 10.24.

### 10.6. The 'bit' Statement

This statement is handled within the "bits" type, see Section 10.53.4.

### 10.7. The 'case' Statement

This statement is mapped to <rng:group> or <rng:interleave> element, depending on whether the statement belongs to an definition of an RPC operation or not. If the argument of a sibling 'default' statement equals to ARGUMENT, @nma:implicit attribute with the value of "true" MUST be added to that <rng:group> or <rng:interleave> element. The @nma:implicit attribute MUST NOT be used for nodes at the top-level of a non-default case (see Section 7.9.3 in [RFC6020]).

### 10.8. The 'choice' Statement

This statement is mapped to <rng:choice> element.

If 'choice' has the 'mandatory' substatement with the value of "true", the attribute @nma:mandatory MUST be added to the <rng:choice> element with the value of ARGUMENT. This case may require additional handling, see Section 11.2.1. Otherwise, if "mandatory true;" is not present, the <rng:choice> element MUST be wrapped in <rng:optional>.

The alternatives in `<rng:choice>` - mapped from either the 'case' statement or a shorthand case - MUST NOT be defined as optional.

#### 10.9. The 'config' Statement

This statement is mapped to `@nma:config` attribute and ARGUMENT becomes its value.

#### 10.10. The 'contact' Statement

This statement SHOULD NOT be used by the mapping since the hybrid schema may be mapped from multiple YANG modules created by different authors. The hybrid schema contains references to all input modules in the Dublin Core elements `<dc:source>`, see Section 10.34. The original YANG modules are the authoritative sources of the authorship information.

#### 10.11. The 'container' Statement

Using the rules specified in Section 9.1.1, the mapping algorithm MUST determine whether the statement defines an optional container, and if so, insert the `<rng:optional>` element and make it the new PARENT.

The container defined by this statement is then mapped to the `<rng:element>` element, which becomes a child of PARENT and uses ARGUMENT with prepended local namespace prefix as the value of its `@name` attribute.

Finally, using the rules specified in Section 9.1.2, the mapping algorithm MUST determine whether the container is implicit, and if so, add the attribute `@nma:implicit` with the value of "true" to the `<rng:element>` element.

#### 10.12. The 'default' Statement

If this statement is a substatement of 'leaf', it is mapped to the `@nma:default` attribute of PARENT and ARGUMENT becomes its value.

As a substatement of 'typedef', the 'default' statement is also mapped to the `@nma:default` attribute with the value of ARGUMENT. The placement of this attribute depends on whether or not the type definition has to be expanded when it is used:

- o If the type definition is not expanded, `@nma:default` becomes an attribute of the `<rng:define>` pattern resulting from the parent 'typedef' mapping.

- o Otherwise, @nma:default becomes an attribute of the ancestor RELAX NG pattern inside which the expansion takes place.

Details and an example are given in Section 9.2.2.

Finally, as a substatement of 'choice', the 'default' statement identifies the default case and is handled within the 'case' statement, see Section 10.7. If the default case uses the shorthand notation where the 'case' statement is omitted, the @nma:implicit attribute with the value of "true" is either attached to the node representing the default case in the shorthand notation or, alternatively, an extra <rng:group> element MAY be inserted and the @nma:implicit attribute attached to it. In the latter case, the net result is the same as if the 'case' statement wasn't omitted for the default case.

EXAMPLE. The following 'choice' statement in a module with namespace prefix "yam"

```
choice leaves {  
  default feuille;  
  leaf feuille { type empty; }  
  leaf hoja { type empty; }  
}
```

is either mapped directly to

```
<rng:choice>  
  <rng:element name="yam:feuille" nma:implicit="true">  
    <rng:empty/>  
  </rng:element>  
  <rng:element name="yam:hoja">  
    <rng:empty/>  
  </rng:element/>  
</rng:choice>
```

or the default case may be wrapped in an extra <rng:group>:

```
<rng:choice>  
  <rng:group nma:implicit="true">  
    <rng:element name="yam:feuille">  
      <rng:empty/>  
    </rng:element>  
  </rng:group>  
  <rng:element name="yam:hoja">  
    <rng:empty/>  
  </rng:element/>  
</rng:choice>
```



#### 10.13. The 'description' Statement

This statement is mapped to the DTD compatibility element `<a:documentation>` and ARGUMENT becomes its text.

In order to get properly formatted in the RELAX NG compact syntax, this element SHOULD be inserted as the first child of PARENT.

#### 10.14. The 'deviation' Statement

This statement is ignored. However, it is assumed that all deviations are known beforehand and the corresponding changes have already been applied to the input YANG modules.

#### 10.15. The 'enum' Statement

This statement is mapped to `<rng:value>` element and ARGUMENT becomes its text. All substatements except 'status' are ignored because the `<rng:value>` element cannot contain annotation elements, see [RNG], section 6.

#### 10.16. The 'error-app-tag' Statement

This statement is ignored unless it is a substatement of 'must'. In the latter case it is mapped to the `<nma:error-app-tag>` element. See also Section 10.35.

#### 10.17. The 'error-message' Statement

This statement is ignored unless it is a substatement of 'must'. In the latter case it is mapped to the `<nma:error-message>` element. See also Section 10.35.

#### 10.18. The 'extension' Statement

This statement is ignored. However, extensions to the YANG language MAY be mapped as described in Section 9.4.

#### 10.19. The 'feature' Statement

This statement is ignored.

#### 10.20. The 'grouping' Statement

This statement is mapped to a RELAX NG named pattern definition `<rng:define>`, but only if the grouping defined by this statement is used without refinements and augments in at least one of the input modules. In this case, the named pattern definition becomes a child

of the `<rng:grammar>` element and its name is ARGUMENT mangled according to the rules specified in Section 9.2.

As explained in Section 8.2, a named pattern definition MUST be placed

- o as a child of the root `<rng:grammar>` element if the corresponding grouping is defined at the top level of an input YANG module;
- o otherwise as a child of the embedded `<rng:grammar>` element corresponding to the module in which the grouping is defined.

Whenever a grouping is used with refinements and/or augments, it is expanded so that the refinements and augments may be applied in place to the prescribed schema nodes. See Section 9.2.1 for further details and an example.

An implementation MAY offer the option of mapping all 'grouping' statements as named pattern definitions in the output RELAX NG schema even if they are not referenced. This is useful for mapping YANG "library" modules that typically contain only 'typedef' and/or 'grouping' statements.

#### 10.21. The 'identity' Statement

This statement is mapped to the following named pattern definition which is placed as a child of the root `<rng:grammar>` element:

```
<rng:define name="__PREFIX_ARGUMENT">
  <rng:choice>
    <rng:value type="QName">PREFIX:ARGUMENT</rng:value>
    <rng:ref name="IDENTITY1"/>
    ...
  </rng:choice>
</rng:define>
```

where

PREFIX is the prefix used in the hybrid schema for the namespace of the module where the current identity is defined.

IDENTITY1 is the name of of the named pattern corresponding to an identity which is derived from the current identity. Exactly one `<rng:ref>` element MUST be present for every such identity.

EXAMPLE ([RFC6020], Section 7.16.3). The identities in the input YANG modules

```
module crypto-base {
  namespace "http://example.com/crypto-base";
  prefix "crypto";
  identity crypto-alg {
    description
      "Base identity from which all crypto algorithms
       are derived.";
  }
}

module des {
  namespace "http://example.com/des";
  prefix "des";
  import "crypto-base" {
    prefix "crypto";
  }
  identity des {
    base "crypto:crypto-alg";
    description "DES crypto algorithm";
  }
  identity des3 {
    base "crypto:crypto-alg";
    description "Triple DES crypto algorithm";
  }
}
```

will be mapped to the following named pattern definitions:

```
<define name="__crypto_crypto-alg">
  <choice>
    <value type="QName">crypto:crypto-alg</value>
    <ref name="__des_des"/>
    <ref name="__des_des3"/>
  </choice>
</define>
<define name="__des_des">
  <value type="QName">des:des</value>
</define>
<define name="__des_des3">
  <value type="QName">des:des3</value>
</define>
```

#### 10.22. The 'if-feature' Statement

ARGUMENT together with arguments of all sibling 'if-feature' statements (with added prefixes, if missing) MUST be collected in a space-separated list which becomes the value of the @nma:if-feature attribute. This attribute is attached to PARENT.

#### 10.23. The 'import' Statement

This statement is not specifically mapped. The module whose name is in ARGUMENT has to be parsed so that the importing module is able to use its top-level groupings, typedefs and identities, and also augment the data tree of the imported module.

If the 'import' statement has the 'revision' substatement, the corresponding revision of the imported module MUST be used. The mechanism for finding a given module revision is outside the scope of this document.

#### 10.24. The 'include' Statement

This statement is not specifically mapped. The submodule whose name is in ARGUMENT has to be parsed and its contents mapped exactly as if the submodule text appeared directly in the main module text.

If the 'include' statement has the 'revision' substatement, the corresponding revision of the submodule MUST be used. The mechanism for finding a given submodule revision is outside the scope of this document.

#### 10.25. The 'input' Statement

This statement is handled within 'rpc' statement, see Section 10.50.

#### 10.26. The 'key' Statement

This statement is mapped to @nma:key attribute. ARGUMENT MUST be translated so that every key is prefixed with the namespace prefix of the local module. The result of this translation then becomes the value of the @nma:key attribute.

#### 10.27. The 'leaf' Statement

This statement is mapped to the <rng:element> element and ARGUMENT with prepended local namespace prefix becomes the value of its @name attribute.

If the leaf is optional, i.e., if there is no "mandatory true;" substatement and the leaf is not declared among the keys of an enclosing list, then the <rng:element> element MUST be enclosed in <rng:optional>, except when the 'leaf' statement is a child of the 'choice' statement and thus represents a shorthand case for that choice (see Section 9.1.1 for details).

### 10.28. The 'leaf-list' Statement

This statement is mapped to a block enclosed by either `<rng:zeroOrMore>` or `<rng:oneOrMore>` element depending on whether the argument of 'min-elements' substatement is "0" or positive, respectively (it is zero by default). This `<rng:zeroOrMore>` or `<rng:oneOrMore>` element becomes the PARENT.

`<rng:element>` is then added as a child element of PARENT and ARGUMENT with prepended local namespace prefix becomes the value of its @name attribute. Another attribute, @nma:leaf-list, MUST also be added to this `<rng:element>` element with the value of "true". If the 'leaf-list' statement has the 'min-elements' substatement and its argument is greater than one, additional attribute @nma:min-elements is attached to `<rng:element>` and the argument of 'min-elements' becomes the value of this attribute. Similarly, if there is the 'max-elements' substatement and its argument value is not "unbounded", attribute @nma:max-elements is attached to this element and the argument of 'max-elements' becomes the value of this attribute.

EXAMPLE. A leaf-list appearing in a module with the namespace prefix "yam"

```
leaf-list foliage {  
    min-elements 3;  
    max-elements 6378;  
    ordered-by user;  
    type string;  
}
```

is mapped to the following RELAX NG fragment:

```
<rng:oneOrMore>  
  <rng:element name="yam:foliage" nma:leaf-list="true"  
    nma:ordered-by="user"  
    nma:min-elements="3" nma:max-elements="6378">  
    <rng:data type="string"/>  
  </rng:element>  
</rng:oneOrMore>
```

### 10.29. The 'length' Statement

This statement is handled within the "string" type, see Section 10.53.10.

## 10.30. The 'list' Statement

This statement is mapped exactly as the 'leaf-list' statement, see Section 10.28. The only difference is that the @nma:leaf-list annotation either MUST NOT be present or MUST have the value of "false".

When mapping the substatements of 'list', the order of children of the list element MUST be specified so that list keys, if there are any, always appear in the same order as they are defined in the 'key' substatement and before other children, see [RFC6020], Section 7.8.5. In particular, if a list key is defined in a grouping but the list node itself is not a part of the same grouping, and the position of the 'uses' statement would violate the above ordering requirement, the grouping MUST be expanded, i.e., the 'uses' statement replaced by the grouping contents.

For example, consider the following YANG fragment of a module with the prefix "yam":

```
grouping keygrp {
  leaf clef {
    type uint8;
  }
}
list foo {
  key clef;
  leaf bar {
    type string;
  }
  leaf baz {
    type string;
  }
  uses keygrp;
}
```

is mapped to the following RELAX NG fragment:

```
<rng:zeroOrMore>
  <rng:element name="yam:foo" nma:key="yam:clef">
    <rng:element name="yam:clef">
      <rng:data type="unsignedByte"/>
    </rng:element>
    <rng:interleave>
      <rng:element name="yam:bar">
        <rng:data type="string"/>
      </rng:element>
      <rng:element name="yam:baz">
        <rng:data type="string"/>
      </rng:element>
    </rng:interleave>
  </rng:element>
</rng:zeroOrMore>
```

Note that the "keygrp" grouping is expanded and the definition of "yam:clef" is moved before the <rng:interleave> pattern.

#### 10.31. The 'mandatory' Statement

This statement may appear as a substatement of 'leaf', 'choice' or 'anyxml' statement. If ARGUMENT is "true", the parent data node is mapped as mandatory, see Section 9.1.1.

As a substatement of 'choice', this statement is also mapped to the @nma:mandatory attribute which is added to PARENT. The value of this attribute is the argument of the parent 'choice' statement.

#### 10.32. The 'max-elements' Statement

This statement is handled within 'leaf-list' or 'list' statements, see Section 10.28.

#### 10.33. The 'min-elements' Statement

This statement is handled within 'leaf-list' or 'list' statements, see Section 10.28.

#### 10.34. The 'module' Statement

This statement is mapped to an embedded <rng:grammar> pattern having the @nma:module attribute with the value of ARGUMENT. In addition, a <dc:source> element SHOULD be created as a child of this <rng:grammar> element and contain ARGUMENT as a metadata reference to the input YANG module. See also Section 10.49.

Substatements of the 'module' statement MUST be mapped so that

- o statements representing configuration/state data are mapped to descendants of the <nma:data> element;
- o statements representing the contents of RPC requests or replies are mapped to descendants of the <nma:rpcs> element;
- o statements representing the contents of event notifications are mapped to descendants of the <nma:notifications> element.

#### 10.35. The 'must' Statement

This statement is mapped to the <nma:must> element. It has one mandatory attribute @assert (with no namespace) which contains ARGUMENT transformed into a valid XPath expression (see Section 9.3). The <nma:must> element may have other subelements resulting from mapping the 'error-app-tag' and 'error-message' substatements. Other substatements of 'must', i.e., 'description' and 'reference', are ignored.

EXAMPLE. YANG statement in the "dhcp" module

```
must 'current() <= ../max-lease-time' {  
    error-message  
        "The default-lease-time must be less than max-lease-time";  
}
```

is mapped to

```
<nma:must assert="current()&lt;=../dhcp:max-lease-time">  
  <nma:error-message>  
    The default-lease-time must be less than max-lease-time  
  </nma:error-message>  
</nma:must>
```

#### 10.36. The 'namespace' Statement

This statement is mapped simultaneously in two ways:

1. To the @xmlns:PREFIX attribute of the root <rng:grammar> element where PREFIX is the namespace prefix specified by the sibling 'prefix' statement. ARGUMENT becomes the value of this attribute.
2. To the @ns attribute of PARENT, which is an embedded <rng:grammar> pattern. ARGUMENT becomes the value of this attribute.



### 10.37. The 'notification' Statement

This statement is mapped to the following subtree of the <nma:notifications> element in the hybrid schema (where PREFIX is the prefix of the local YANG module):

```
<nma:notification>
  <rng:element name="PREFIX:ARGUMENT">
    ...
  </rng:element>
</nma:notification>
```

Substatements of 'notification' are mapped under <rng:element name="PREFIX:ARGUMENT">.

### 10.38. The 'ordered-by' Statement

This statement is mapped to @nma:ordered-by attribute and ARGUMENT becomes the value of this attribute. See Section 10.28 for an example.

### 10.39. The 'organization' Statement

This statement is ignored by the mapping because the hybrid schema may be mapped from multiple YANG modules authored by different parties. The hybrid schema SHOULD contain references to all input modules in the Dublin Core <dc:source> elements, see Section 10.34. The original YANG modules are the authoritative sources of the authorship information.

### 10.40. The 'output' Statement

This statement is handled within the 'rpc' statement, see Section 10.50.

### 10.41. The 'path' Statement

This statement is handled within the "leafref" type, see Section 10.53.8.

### 10.42. The 'pattern' Statement

This statement is handled within the "string" type, see Section 10.53.10.

## 10.43. The 'position' Statement

This statement is ignored.

## 10.44. The 'prefix' Statement

This statement is handled within the sibling 'namespace' statement, see Section 10.36, or within the parent 'import' statement, see Section 10.23. As a substatement of 'belongs-to' (in submodules), the 'prefix' statement is ignored.

## 10.45. The 'presence' Statement

This statement influences the mapping of the parent container (Section 10.11): the parent container definition MUST be wrapped in `<rng:optional>`, regardless of its contents. See also Section 9.1.1.

## 10.46. The 'range' Statement

This statement is handled within numeric types, see Section 10.53.9.

## 10.47. The 'reference' Statement

This statement is mapped to `<a:documentation>` element and its text is set to ARGUMENT prefixed with "See: ".

## 10.48. The 'require-instance' Statement

This statement is handled within "instance-identifier" type (Section 10.53.7).

## 10.49. The 'revision' Statement

The mapping uses only the most recent instance of the 'revision' statement, i.e., one with the latest date in ARGUMENT, which specifies the current revision of the input YANG module [RFC6020]. This date SHOULD be recorded, together with the name of the YANG module, in the corresponding Dublin Core `<dc:source>` element (see Section 10.34), for example in this form:

```
<dc:source>YANG module 'foo', revision 2010-03-02</dc:source>
```

The 'description' substatement of 'revision' is ignored.

## 10.50. The 'rpc' Statement

This statement is mapped to the following subtree in the RELAX NG schema (where PREFIX is the prefix of the local YANG module):

```
<nma:rpc>
  <nma:input>
    <rng:element name="PREFIX:ARGUMENT">
      ... mapped contents of 'input' ...
    </rng:element>
  </nma:input>
  <nma:output">
    ... mapped contents of 'output' ...
  </nma:output>
</nma:rpc>
```

As indicated in the schema fragment, contents of the 'input' substatement (if any) are mapped under `<rng:element name="PREFIX:ARGUMENT">`. Similarly, contents of the 'output' substatement are mapped under `<nma:output">`. If there is no 'output' substatement, the `<nma:output>` element MUST NOT be present.

The `<nma:rpc>` element is a child of `<nma:rpcs>`.

#### 10.51. The 'status' Statement

This statement MAY be ignored. Otherwise, it is mapped to `@nma:status` attribute and `ARGUMENT` becomes its value.

#### 10.52. The 'submodule' Statement

This statement is not specifically mapped. Its substatements are mapped as if they appeared directly in the module the submodule belongs to.

#### 10.53. The 'type' Statement

Most YANG built-in datatypes have an equivalent in the XSD datatype library [XSD-D] as shown in Table 4.

YANG type	XSD type	Meaning
int8	byte	8-bit integer value
int16	short	16-bit integer value
int32	int	32-bit integer value
int64	long	64-bit integer value
uint8	unsignedByte	8-bit unsigned integer value
uint16	unsignedShort	16-bit unsigned integer value
uint32	unsignedInt	32-bit unsigned integer value
uint64	unsignedLong	64-bit unsigned integer value
string	string	character string
binary	base64Binary	binary data in base64 encoding

Table 4: YANG built-in datatypes with equivalents in the W3C XML Schema Type Library

Two important datatypes of the XSD datatype library - "dateTime" and "anyURI" - are not built-in types in YANG but instead are defined as derived types in the standard modules [RFC6021]: "date-and-time" in the "ietf-yang-types" module and "uri" in the "ietf-inet-types" module. However, the formal restrictions in the YANG type definitions are rather weak. Therefore, implementations of the YANG-to-DSDL mapping SHOULD detect these derived types in source YANG modules and map them to "dateTime" and "anyURI", respectively.

Details about the mapping of individual YANG built-in types are given in the following subsections.

#### 10.53.1. The "empty" Type

This type is mapped to `<rng:empty/>`.

#### 10.53.2. The "boolean" Type

This built-in type does not allow any restrictions and is mapped to the following XML fragment:

```
<rng:choice>
  <rng:value>true</rng:value>
  <rng:value>>false</rng:value>
</rng:choice>
```

Note that the XSD "boolean" type cannot be used here because it allows, unlike YANG, an alternative numeric representation of boolean values: 0 for "false" and 1 for "true".

#### 10.53.3. The "binary" Type

This built-in type does not allow any restrictions and is mapped simply by inserting `<rng:data>` element whose `@type` attribute value is set to "base64Binary" (see also Table 4).

#### 10.53.4. The "bits" Type

This type is mapped to `<rng:list>` and for each 'bit' substatement the following XML fragment is inserted as a child of `<rng:list>`:

```
<rng:optional>
  <rng:value>bit_name</rng:value>
</rng:optional>
```

where `bit_name` is the name of the bit as found in the argument of a 'bit' substatement.

#### 10.53.5. The "enumeration" and "union" Types

These types are mapped to the `<rng:choice>` element.

#### 10.53.6. The "identityref" Type

This type is mapped to the following named pattern reference:

```
<rng:ref name="__PREFIX_BASE"/>
```

where `PREFIX:BASE` is the qualified name of the identity appearing in the argument of the 'base' substatement.

For example, assume that module "des" in Section 10.21 contains the following leaf definition:

```
leaf foo {
  type identityref {
    base crypto:crypto-alg;
  }
}
```

This leaf would then be mapped to the following element pattern:

```
<element name="des:foo">
  <ref name="__crypto_crypto-alg"/>
</element>
```

#### 10.53.7. The "instance-identifier" Type

This type is mapped to `<rng:data>` element with `@type` attribute set to "string". In addition, an empty `<nma:instance-identifier>` element MUST be inserted as a child of PARENT.

The argument of the 'require-instance' substatement, if it exists, becomes the value of the `@require-instance` attribute of the `<nma:instance-identifier>` element.

#### 10.53.8. The "leafref" Type

This type is mapped exactly as the type of the leaf given in the argument of 'path' substatement. However, if the type of the referred leaf defines a default value, this default value MUST be ignored by the mapping.

In addition, `@nma:leafref` attribute MUST be added to PARENT. The argument of the 'path' substatement, translated according to Section 9.3, is set as the value of this attribute.

#### 10.53.9. The Numeric Types

YANG built-in numeric types are "int8", "int16", "int32", "int64", "uint8", "uint16", "uint32", "uint64" and "decimal64". They are mapped to `<rng:data>` element with `@type` attribute set to ARGUMENT translated according to Table 4 above.

An exception is the "decimal64" type, which is mapped to the "decimal" type of the XSD datatype library. Its precision and number of fractional digits are controlled with the following facets, which MUST always be present:

- o "totalDigits" facet set to the value of 19.
- o "fractionDigits" facet set to the argument of the 'fraction-digits' substatement.

The fixed value of "totalDigits" corresponds to the maximum of 19 decimal digits for 64-bit integers.

For example, the statement

```
type decimal64 {  
    fraction-digits 2;  
}
```

is mapped to the following RELAX NG datatype:

```
<rng:data type="decimal">  
    <rng:param name="totalDigits">19</rng:param>  
    <rng:param name="fractionDigits">2</rng:param>  
</rng:data>
```

All numeric types support the 'range' restriction, which is mapped as follows:

If the range expression consists of just a single range LO..HI, then it is mapped to a pair of datatype facets

```
<rng:param name="minInclusive">LO</rng:param>
```

and

```
<rng:param name="maxInclusive">HI</rng:param>
```

If the range consists of a single number, the values of both facets are set to this value. If LO is equal to the string "min", the "minInclusive" facet is omitted. If HI is equal to the string "max", the "maxInclusive" facet is omitted.

If the range expression has multiple parts separated by "|", then the parent <rng:data> element must be repeated once for every range part and all such <rng:data> elements are wrapped in <rng:choice> element. Each <rng:data> element contains the "minInclusive" and "maxInclusive" facets for one part of the range expression as described in the previous paragraph.

For the "decimal64" type, the "totalDigits" and "fractionDigits" must be repeated inside each of the <rng:data> elements.

For example,

```
type int32 {  
    range "-6378..0|42|100..max";  
}
```

is mapped to the following RELAX NG fragment:

```
<rng:choice>
  <rng:data type="int">
    <rng:param name="minInclusive">-6378</rng:param>
    <rng:param name="maxInclusive">0</rng:param>
  </rng:data>
  <rng:data type="int">
    <rng:param name="minInclusive">42</rng:param>
    <rng:param name="maxInclusive">42</rng:param>
  </rng:data>
  <rng:data type="int">
    <rng:param name="minInclusive">100</rng:param>
  </rng:data>
</rng:choice>
```

See Section 9.2.2 for further details on mapping the restrictions.

#### 10.53.10. The "string" Type

This type is mapped to `<rng:data>` element with the `@type` attribute set to "string".

The 'length' restriction is handled analogically to the 'range' restriction for the numeric types (Section 10.53.9):

If the length expression has just a single range, then

- o if the length range consists of a single number LENGTH, the following datatype facet is inserted:

```
<rng:param name="length">LENGTH</rng:param>.
```

- o Otherwise the length range is of the form LO..HI, i.e., it consists of both the lower and upper bound. The following two datatype facets are then inserted:

```
<rng:param name="minLength">LO</rng:param>
```

and

```
<rng:param name="maxLength">HI</rng:param>
```

If LO is equal to the string "min", the "minLength" facet is omitted. If HI is equal to the string "max", the "maxLength" facet is omitted.

If the length expression has of multiple parts separated by "|", then the parent `<rng:data>` element must be repeated once for every range part and all such `<rng:data>` elements are wrapped in `<rng:choice>`



element. Each `<rng:data>` element contains the "length" or "minLength" and "maxLength" facets for one part of the length expression as described in the previous paragraph.

Every 'pattern' restriction of the "string" datatype is mapped to the "pattern" facet

```
<rng:param name="pattern">...</rng:param>
```

with text equal to the argument of the 'pattern' statement. All such "pattern" facets must be repeated inside each copy of the `<rng:data>` element, i.e., once for each length range.

For example,

```
type string {  
    length "1|3..8";  
    pattern "[A-Z][a-z]*";  
}
```

is mapped to the following RELAX NG fragment:

```
<rng:choice>  
  <rng:data type="string">  
    <rng:param name="length">1</rng:param>  
    <rng:param name="pattern">[A-Z][a-z]*</rng:param>  
  </rng:data>  
  <rng:data type="string">  
    <rng:param name="minLength">3</rng:param>  
    <rng:param name="maxLength">8</rng:param>  
    <rng:param name="pattern">[A-Z][a-z]*</rng:param>  
  </rng:data>  
</rng:choice>
```

#### 10.53.11. Derived Types

If the 'type' statement refers to a derived type, it is mapped in one of the following ways depending on whether it contains any restrictions as its substatements:

1. Without restrictions, the 'type' statement is mapped simply to the `<rng:ref>` element, i.e., a reference to a named pattern. If the RELAX NG definition of this named pattern has not been added to the hybrid schema yet, the corresponding type definition **MUST** be found and its mapping installed as a subelement of either the root or an embedded `<rng:grammar>` element, see Section 10.54. Even if a given derived type is used more than once in the input YANG modules, the mapping of the corresponding 'typedef' **MUST** be

installed only once.

2. If any restrictions are present, the ancestor built-in type for the given derived type must be determined and the mapping of this base type MUST be used. Restrictions appearing at all stages of the type derivation chain MUST be taken into account and their conjunction added to the `<rng:data>` element which defines the basic type.

See Section 9.2.2 for more details and an example.

#### 10.54. The 'typedef' Statement

This statement is mapped to a RELAX NG named pattern definition `<rng:define>`, but only if the type defined by this statement is used without restrictions in at least one of the input modules. In this case, the named pattern definition becomes a child of either the root or an embedded `<rng:grammar>` element, depending on whether the 'typedef' statement appears at the top level of a YANG module or not. The name of this named pattern definition is set to ARGUMENT mangled according to the rules specified in Section 9.2.

Whenever a derived type is used with additional restrictions, the ancestor built-in type for the derived type is used instead with restrictions (facets) that are a combination of all restrictions specified along the type derivation chain. See Section 10.53.11 for further details and an example.

An implementation MAY offer the option of recording all 'typedef' statements as named patterns in the output RELAX NG schema even if they are not referenced. This is useful for mapping YANG "library" modules containing only 'typedef' and/or 'grouping' statements.

#### 10.55. The 'unique' Statement

This statement is mapped to `@nma:unique` attribute. ARGUMENT MUST be translated so that every node identifier in each of its components is prefixed with the namespace prefix of the local module, unless the prefix is already present. The result of this translation then becomes the value of the `@nma:unique` attribute.

For example, assuming that the local module prefix is "ex",

```
unique "foo ex:bar/baz"
```

is mapped to the following attribute/value pair:

```
nma:unique="ex:foo ex:bar/ex:baz"
```

## 10.56. The 'units' Statement

This statement is mapped to @nma:units attribute and ARGUMENT becomes its value.

## 10.57. The 'uses' Statement

If this statement has neither 'refine' nor 'augment' substatements, it is mapped to <rng:ref> element, i.e., a reference to a named pattern, and the value of its @name attribute is set to ARGUMENT mangled according to Section 9.2. If the RELAX NG definition of the referenced named pattern has not been added to the hybrid schema yet, the corresponding grouping MUST be found and its mapping installed as a subelement of <rng:grammar>, see Section 10.20.

Otherwise, if the 'uses' statement has any 'refine' or 'augment' substatements, the corresponding grouping must be looked up and its contents inserted under PARENT. See Section 9.2.1 for further details and an example.

## 10.58. The 'value' Statement

This statement is ignored.

## 10.59. The 'when' Statement

This statement is mapped to @nma:when attribute and ARGUMENT, translated according to Section 9.3, becomes its value.

## 10.60. The 'yang-version' Statement

This statement is not mapped to the output schema. However, an implementation SHOULD check that it is compatible with the YANG version declared by the statement (currently version 1). In the case of a mismatch, the implementation SHOULD report an error and terminate.

## 10.61. The 'yin-element' Statement

This statement is not mapped to the output schema, but see the rules for extension handling in Section 9.4.

## 11. Mapping the Hybrid Schema to DSDL

As explained in Section 6, the second step of the YANG-to-DSDL mapping takes the hybrid schema and transforms it to various DSDL schemas capable of validating instance XML documents. As an input parameter, this step takes, in the simplest case, just a specification of the NETCONF XML document type that is to be validated. These document types can be, for example, the contents of a datastore, a reply to `<nc:get>` or `<nc:get-config>`, contents of other RPC requests/replies and event notifications, and so on.

The second mapping step has to accomplish the following three general tasks:

1. Extract the parts of the hybrid schema that are appropriate for the requested document type. For example, if a `<nc:get>` reply is to be validated, the subtree under `<nma:data>` has to be selected.
2. The schema must be adapted to the specific encapsulating XML elements mandated by the RPC layer. These are, for example, `<nc:rpc>` and `<nc:data>` elements in the case of a `<nc:get>` reply or `<en:notification>` for a notification.
3. Finally, NETMOD-specific annotations that are relevant for the schema language of the generated schema must be mapped to the corresponding patterns or rules.

These three tasks are together much simpler than the first mapping step and can be effectively implemented using XSL transformations [XSLT].

The following subsections describe the details of the second mapping step for the individual DSDL schema languages. Section 12 then contains a detailed specification for the mapping of all NETMOD-specific annotations.

### 11.1. Generating RELAX NG Schemas for Various Document Types

With one minor exception, obtaining a validating RELAX NG schema from the hybrid schema only means taking appropriate parts of the hybrid schema and assembling them in a new RELAX NG grammar, perhaps after removing all unwanted annotations.

The structure of the resulting RELAX NG schema is similar to that of the hybrid schema: The root grammar contains embedded grammars, one for each input YANG module. However, as explained in Section 8.2, global named pattern definitions (children of the root `<rng:grammar>` element) MUST be moved to a separate schema file.

Depending on the XML document type that is the target for validation, such as <nc:get>/<nc:get-config> reply, RPC operations or notifications, patterns defining corresponding top-level information items MUST be added, such as <nc:rpc-reply> with the @message-id attribute and so on.

In order to avoid copying common named pattern definitions for common NETCONF elements and attributes to every single output RELAX NG file, such schema-independent definitions SHOULD be collected in a library file which is then included by the validating RELAX NG schemas. Appendix B has the listing of such a library file.

The minor exception mentioned above is the annotation @nma:config, which must be observed if the target document type is a reply to <nc:get-config>. In this case, each element definition that has this attribute with the value of "false" MUST be removed from the schema together with its descendants. See Section 12.1 for more details.

#### 11.2. Mapping Semantic Constraints to Schematron

Schematron schemas tend to be much flatter and more uniform compared to RELAX NG. They have exactly four levels of XML hierarchy: <sch:schema>, <sch:pattern>, <sch:rule> and <sch:assert> or <sch:report>.

In a Schematron schema generated by the second mapping step, the basic unit of organization is a rule represented by the <sch:rule> element. The following NETMOD-specific annotations from the hybrid schema (henceforth called "semantic annotations") are mapped to corresponding Schematron rules: <nma:must>, @nma:key, @nma:unique, @nma:max-elements, @nma:min-elements, @nma:when, @nma:leafref, @nma:leaf-list, and also @nma:mandatory appearing as an attribute of <rng:choice> (see Section 11.2.1).

Each input YANG module is mapped to a Schematron pattern whose @id attribute is set to the module name. Every <rng:element> pattern containing at least one of the above-mentioned semantic annotations is then mapped to a Schematron rule:

```
<sch:rule context="XELEM">
  ...
</sch:rule>
```

The value of the mandatory @context attribute of <sch:rule> (denoted as XELEM) MUST be set to the absolute path of the context element in the data tree. The <sch:rule> element contains the mappings of all contained semantic annotations in the form of Schematron asserts or reports.

Semantic annotations appearing inside a named pattern definition (i.e., having `<rng:define>` among its ancestors) require special treatment because they may be potentially used in different contexts. This is accomplished by using Schematron abstract patterns that use the `"$pref"` variable in place of the local namespace prefix. The value of the `@id` attribute of such an abstract pattern MUST be set to the name of the named pattern definition which is being mapped (i.e., the mangled name of the original YANG grouping).

When the abstract pattern is instantiated, the values of the following two parameters MUST be provided:

- o `pref`: the actual namespace prefix,
- o `start`: XPath expression defining the context in which the grouping is used.

EXAMPLE. Consider the following YANG module:

```
module example4 {  
  namespace "http://example.com/ns/example4";  
  prefix ex4;  
  uses sorted-leaf-list;  
  grouping sorted-leaf-list {  
    leaf-list sorted-entry {  
      must "not(preceding-sibling::sorted-entry > .)" {  
        error-message "Entries must appear in ascending order.";  
      }  
      type uint8;  
    }  
  }  
}
```

The resulting Schematron schema for a reply to `<nc:get>` is then as follows:

```

<?xml version="1.0" encoding="utf-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <sch:ns uri="http://example.com/ns/example4" prefix="ex4"/>
  <sch:ns uri="urn:ietf:params:xml:ns:netconf:base:1.0"
    prefix="nc"/>
  <sch:pattern abstract="true"
    id="_example4__sorted-leaf-list">
    <sch:rule context="$start/$pref:sorted-entry">
      <sch:report
        test=". = preceding-sibling::$pref:sorted-entry">
        Duplicate leaf-list entry "<sch:value-of select="."/>".
      </sch:report>
      <sch:assert
        test="not(preceding-sibling::$pref:sorted-entry > .)">
        Entries must appear in ascending order.
      </sch:assert>
    </sch:rule>
  </sch:pattern>
  <sch:pattern id="example4"/>
  <sch:pattern id="id2573371" is-a="_example4__sorted-leaf-list">
    <sch:param name="start" value="/nc:rpc-reply/nc:data"/>
    <sch:param name="pref" value="ex4"/>
  </sch:pattern>
</sch:schema>

```

The "sorted-leaf-list" grouping from the input module is mapped to an abstract pattern with an @id value of "\_example4\_\_sorted-leaf-list" in which the 'must' statement corresponds to the <sch:assert> element. The abstract pattern is instantiated by the pattern with an @id value of "id2802112" which sets the "start" and "pref" parameters to appropriate values.

Note that another Schematron element, <sch:report>, was automatically added, checking for duplicate leaf-list entries.

The mapping from the hybrid schema to Schematron proceeds in the following steps:

1. First, the active subtree(s) of the hybrid schema must be selected depending on the requested target document type. This procedure is identical to the RELAX NG case, including the handling of @nma:config if the target document type is <nc:get-config> reply.
2. Namespaces of all input YANG modules, together with the namespaces of base NETCONF ("nc" prefix) or notifications ("en" prefix) MUST be declared using the <sch:ns> element, for example

```
<sch:ns uri="http://example.com/ns/example4" prefix="ex4"/>
```

3. One pattern is created for every input module. In addition, an abstract pattern is created for every named pattern definition containing one or more semantic annotations.
4. A <sch:rule> element is created for each element pattern containing semantic annotations.
5. Every such annotation is then mapped to an <sch:assert> or <sch:report> element which is installed as a child of the <sch:rule> element.

#### 11.2.1. Constraints on Mandatory Choice

In order to fully represent the semantics of YANG's 'choice' statement with the "mandatory true;" substatement, the RELAX NG grammar has to be combined with a special Schematron rule.

EXAMPLE. Consider the following module:

```
module example5 {  
  namespace "http://example.com/ns/example5";  
  prefix ex5;  
  choice foobar {  
    mandatory true;  
    case foo {  
      leaf foo1 {  
        type uint8;  
      }  
      leaf foo2 {  
        type uint8;  
      }  
    }  
    leaf bar {  
      type uint8;  
    }  
  }  
}
```

In this module, all three leaf nodes in both case branches are optional but because of the "mandatory true;" statement, at least one of them must be present in a valid configuration. The 'choice' statement from this module is mapped to the following fragment of the RELAX NG schema:



```
<rng:choice>
  <rng:interleave>
    <rng:optional>
      <rng:element name="ex5:foo1">
        <rng:data type="unsignedByte"/>
      </rng:element>
    </rng:optional>
    <rng:optional>
      <rng:element name="ex5:foo2">
        <rng:data type="unsignedByte"/>
      </rng:element>
    </rng:optional>
  </rng:interleave>
  <rng:element name="ex5:bar">
    <rng:data type="unsignedByte"/>
  </rng:element>
</rng:choice>
```

In the second case branch, the "ex5:bar" element is defined as mandatory so that this element must be present in a valid configuration if this branch is selected. However, the two elements in the first branch "foo" cannot be both declared as mandatory since each of them alone suffices for a valid configuration. As a result, the above RELAX NG fragment would successfully validate configurations where none of the three leaf elements are present.

Therefore, mandatory choices, which can be recognized in the hybrid schema as <rng:choice> elements with the @nma:mandatory annotation, have to be handled in a special way: For each mandatory choice where at least one of the cases contains more than one node, a Schematron rule MUST be added enforcing the presence of at least one element from any of the cases. (RELAX NG schema guarantees that elements from different cases cannot be mixed together, that all mandatory nodes are present etc.).

For the example module above, the Schematron rule will be as follows:

```
<sch:rule context="/nc:rpc-reply/nc:data">
  <sch:assert test="ex5:foo1 or ex5:foo2 or ex5:bar">
    Node(s) from at least one case of choice "foobar" must exist.
  </sch:assert>
</sch:rule>
```

### 11.3. Mapping Default Values to DSRL

DSRL is the only component of DSDL which is allowed to change the information set of the validated XML document. While DSRL also has other functions, YANG-to-DSDL mapping uses it only for specifying and

applying default contents. For XML instance documents based on YANG data models, insertion of default contents may potentially take place for all implicit nodes identified by the rules in Section 9.1.2.

In DSRL, the default contents of an element are specified using the `<dsrl:default-content>` element, which is a child of `<dsrl:element-map>`. Two sibling elements of `<dsrl:default-content>` determine the context for the application of the default contents, see [DSRL]:

- o `<dsrl:parent>` element contains an XSLT pattern specifying the parent element; the default contents are applied only if the parent element exists in the instance document.
- o `<dsrl:name>` contains the XML name of the element which, if missing or empty, is inserted together with the contents of `<dsrl:default-content>`.

The `<dsrl:parent>` element is optional in a general DSRL schema but, for the purpose of the YANG-to-DSDL mapping, this element **MUST** be always present, in order to guarantee a proper application of default contents.

DSRL mapping only deals with `<rng:element>` patterns in the hybrid schema that define implicit nodes (see Section 9.1.2). Such element patterns are distinguished by having NETMOD-specific annotation attributes `@nma:default` or `@nma:implicit`, i.e., either

```
<rng:element name="ELEM" nma:default="DEFVALUE">
  ...
</rng:element>
```

or

```
<rng:element name="ELEM" nma:implicit="true">
  ...
</rng:element>
```

The former case applies to leaf nodes having the 'default' substatement, but also to leaf nodes that obtain their default value from a typedef, if this typedef is expanded according to the rules in Section 9.2.2 so that the `@nma:default` annotation is attached directly to the leaf's element pattern.

The latter case is used for all implicit containers (see Section 9.1) and for leaves that obtain the default value from a typedef and don't have the `@nma:default` annotation.

In the simplest case, both element patterns are mapped to the

following DSRL element map:

```
<dsrl:element-map>
  <dsrl:parent>XPARENT</dsrl:parent>
  <dsrl:name>ELEM</dsrl:name>
  <dsrl:default-content>DEFCONT</dsrl:default-content>
</dsrl:element-map>
```

where XPARENT is the absolute XPath of ELEM's parent element in the data tree and DEFCONT is constructed as follows:

- o If the implicit node ELEM is a leaf and has the @nma:default attribute, DEFCONT is set to the value of this attribute (denoted above as DEFVALUE).
- o If the implicit node ELEM is a leaf and has the @nma:implicit attribute with the value of "true", the default value has to be determined from the @nma:default attribute of the definition of ELEM's type (perhaps recursively) and used in place of DEFCONT in the above DSRL element map. See also Section 9.2.2.
- o Otherwise, the implicit node ELEM is a container and DEFCONT is constructed as an XML fragment containing all descendant elements of ELEM that have either @nma:implicit or @nma:default attribute.

In addition, when mapping the default case of a choice, it has to be guaranteed that the default contents are not applied if any node from any non-default case is present. This is accomplished by setting <dsrl:parent> in a special way:

```
<dsrl:parent>XPARENT[not (ELEM1|ELEM2|...|ELEMn)]</dsrl:parent>
```

where ELEM1, ELEM2, ... ELEMn are the names of all top-level nodes from all non-default cases. The rest of the element map is exactly as before.

EXAMPLE. Consider the following YANG module:

```
module example6 {  
  namespace "http://example.com/ns/example6";  
  prefix ex6;  
  container outer {  
    leaf leaf1 {  
      type uint8;  
      default 1;  
    }  
    choice one-or-two {  
      default "one";  
      container one {  
        leaf leaf2 {  
          type uint8;  
          default 2;  
        }  
      }  
      leaf leaf3 {  
        type uint8;  
        default 3;  
      }  
    }  
  }  
}
```

The DSRL schema generated for the "get-reply" target document type will be:

```
<?xml version="1.0" encoding="utf-8"?>
<dsrl:maps xmlns:dsrl="http://purl.oclc.org/dsdl/dsrl"
           xmlns:ex6="http://example.com/ns/example6"
           xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data</dsrl:parent>
    <dsrl:name>ex6:outer</dsrl:name>
    <dsrl:default-content>
      <ex6:leaf1>1</ex6:leaf1>
      <ex6:one>
        <ex6:leaf2>2</ex6:leaf2>
      </ex6:one>
    </dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data/ex6:outer</dsrl:parent>
    <dsrl:name>ex6:leaf1</dsrl:name>
    <dsrl:default-content>1</dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>
      /nc:rpc-reply/nc:data/ex6:outer[not(ex6:leaf3)]
    </dsrl:parent>
    <dsrl:name>ex6:one</dsrl:name>
    <dsrl:default-content>
      <ex6:leaf2>2</ex6:leaf2>
    </dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>
      /nc:rpc-reply/nc:data/ex6:outer/ex6:one
    </dsrl:parent>
    <dsrl:name>ex6:leaf2</dsrl:name>
    <dsrl:default-content>2</dsrl:default-content>
  </dsrl:element-map>
</dsrl:maps>
```

Note that the default value for "leaf3" defined in the YANG module is ignored because "leaf3" represents a non-default alternative of a choice and as such never becomes an implicit element.

## 12. Mapping NETMOD-specific Annotations to DSDL Schema Languages

This section contains the mapping specification for the individual NETMOD-specific annotations. In each case, the result of the mapping must be inserted into an appropriate context of the target DSDL schema as described in Section 11. The context is determined by the element pattern in the hybrid schema to which the annotation is attached. In the rest of this section, CONTELEM will denote the name of this context element properly qualified with its namespace prefix.

### 12.1. The @nma:config Annotation

If this annotation is present with the value of "false", the following rules MUST be observed for DSDL schemas of <nc:get-config> reply:

- o When generating RELAX NG, the contents of the CONTELEM definition MUST be changed to <rng:notAllowed>.
- o When generating Schematron or DSRL, the CONTELEM definition and all its descendants in the hybrid schema MUST be ignored.

### 12.2. The @nma:default Annotation

This annotation is used for generating the DSRL schema as described in Section 11.3.

### 12.3. The <nma:error-app-tag> Annotation

This annotation currently has no mapping defined.

### 12.4. The <nma:error-message> Annotation

This annotation is handled within <nma:must>, see Section 12.13.

### 12.5. The @if-feature Annotation

The information about available features MAY be supplied as an input parameter to an implementation. In this case, the following changes MUST be performed for all features that are considered unavailable:

- o When generating RELAX NG, the contents of the CONTELEM definition MUST be changed to <rng:notAllowed>.
- o When generating Schematron or DSRL, the CONTELEM definition and all its descendants in the hybrid schema MUST be ignored.

#### 12.6. The @nma:implicit Annotation

This annotation is used for generating the DSRL schema as described in Section 11.3.

#### 12.7. The <nma:instance-identifier> Annotation

If this annotation element has the @require-instance attribute with the value of "false", it is ignored. Otherwise it is mapped to the following Schematron assert:

```
<sch:assert test="nmf:evaluate(.)">
  The element pointed to by "CONTELEM" must exist.
</sch:assert>
```

The nmf:evaluate() function is an XSLT extension function (see Extension Functions in [XSLT]) that evaluates an XPath expression at run time. Such an extension function is available in Extended XSLT (EXSLT) or provided as a proprietary extension by some XSLT processors, for example Saxon.

#### 12.8. The @nma:key Annotation

Assume this annotation attribute contains "k\_1 k\_2 ... k\_n", i.e., specifies n children of CONTELEM as list keys. The annotation is then mapped to the following Schematron report:

```
<sch:report test="CONDITION">
  Duplicate key of list "CONTELEM"
</sch:report>
```

where CONDITION has this form:

preceding-sibling::CONTELEM[C\_1 and C\_2 and ... and C\_n]

Each sub-expression C\_i, for i=1,2,...,n, specifies the condition for violated uniqueness of the key k\_i, namely

k\_i=current()/k\_i

#### 12.9. The @nma:leaf-list Annotation

This annotation is mapped to the following Schematron rule which detects duplicate entries of a leaf-list:

```
<sch:report
  test=". = preceding-sibling::PREFIX:sorted-entry">
  Duplicate leaf-list entry "<sch:value-of select="."/>".
</sch:report>
```

See Section 11.2 for a complete example.

#### 12.10. The @nma:leafref Annotation

This annotation is mapped to the following assert:

```
<sch:assert test="PATH=.">  
  Leaf "PATH" does not exist for leafref value "VALUE"  
</sch:assert>
```

where PATH is the value of @nma:leafref and VALUE is the value of the context element in the instance document for which the referred leaf doesn't exist.

#### 12.11. The @nma:min-elements Annotation

This annotation is mapped to the following Schematron assert:

```
<sch:assert test="count(../CONTELEM)>=MIN">  
  List "CONTELEM" - item count must be at least MIN  
</sch:assert>
```

where MIN is the value of @nma:min-elements.

#### 12.12. The @nma:max-elements Annotation

This annotation is mapped to the following Schematron assert:

```
<sch:assert  
  test="count(../CONTELEM)<=MAX or preceding-sibling:../CONTELEM">  
  Number of list items must be at most MAX  
</sch:assert>
```

where MAX is the value of @nma:min-elements.

#### 12.13. The <nma:must> Annotation

This annotation is mapped to the following Schematron assert:

```
<sch:assert test="EXPRESSION">  
  MESSAGE  
</sch:assert>
```

where EXPRESSION is the value of the mandatory @assert attribute of <nma:must>. If the <nma:error-message> subelement exists, MESSAGE is set to its contents, otherwise it is set to the default message "Condition EXPRESSION must be true".



## 12.14. The &lt;nma:ordered-by&gt; Annotation

This annotation currently has no mapping defined.

## 12.15. The &lt;nma:status&gt; Annotation

This annotation currently has no mapping defined.

## 12.16. The @nma:unique Annotation

The mapping of this annotation is almost identical as for @nma:key, see Section 12.8, with two small differences:

- o The value of @nma:unique is a list of descendant schema node identifiers rather than simple leaf names. However, the XPath expressions specified in Section 12.8 work without any modifications if the descendant schema node identifiers are substituted for k\_1, k\_2, ..., k\_n.
- o The message appearing as the text of <sch:report> is different: "Violated uniqueness for list CONTELEM".

## 12.17. The @nma:when Annotation

This annotation is mapped to the following Schematron assert:

```
<sch:assert test="EXPRESSION">  
  Node "CONTELEM" is only valid when "EXPRESSION" is true.  
</sch:assert>
```

where EXPRESSION is the value of @nma:when.

## 13. IANA Considerations

This document requests the following two registrations of namespace URIs in the IETF XML registry [RFC3688]:

-----  
URI: urn:ietf:params:xml:ns:netmod:dSDL-annotations:1

Registrant Contact: The IESG.

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

-----  
URI: urn:ietf:params:xml:ns:netmod:xpath-extensions:1

Registrant Contact: The IESG.

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

#### 14. Security Considerations

This document defines a procedure that maps data models expressed in the YANG language to a coordinated set of DSDL schemas. The procedure itself has no security impact on the Internet.

DSDL schemas obtained by the mapping procedure may be used for validating the contents of NETCONF messages or entire datastores and thus provide additional validity checks above those performed by NETCONF server and client implementations supporting YANG data models. The strictness of this validation is directly derived from the source YANG modules that the validated XML data adhere to.

## 15. Contributors

The following people contributed significantly to the initial version of this document:

- o Rohan Mahy
- o Sharon Chisholm (Ciena)

## 16. Acknowledgments

The editor wishes to thank the following individuals who provided helpful suggestions and/or comments on various versions of this document: Andy Bierman, Martin Bjorklund, Jirka Kosek, Juergen Schoenwaelder and Phil Shafer.

## 17. References

### 17.1. Normative References

- [DSDL] ISO/IEC, "Document Schema Definition Languages (DSDL) - Part 1: Overview", ISO/IEC 19757-1, November 2004.
- [DSRL] ISO/IEC, "Information Technology - Document Schema Definition Languages (DSDL) - Part 8: Document Semantics Renaming Language - DSRL", ISO/IEC 19757-8:2008(E), December 2008.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for Network Configuration Protocol (NETCONF)", RFC 6020, September 2010.
- [RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, September 2010.
- [RNG] ISO/IEC, "Information Technology - Document Schema Definition Languages (DSDL) - Part 2: Regular-Grammar-Based Validation - RELAX NG. Second Edition.", ISO/IEC 19757-2:2008(E), December 2008.
- [RNG-CS] ISO/IEC, "Information Technology - Document Schema Definition Languages (DSDL) - Part 2: Regular-Grammar-Based Validation - RELAX NG. AMENDMENT 1: Compact Syntax", ISO/IEC 19757-2:2003/Amd. 1:2006(E), January 2006.
- [RNG-DTD] Clark, J., Ed. and M. Murata, Ed., "RELAX NG DTD Compatibility", OASIS Committee Specification 3 December 2001, December 2001.
- [Schematron] ISO/IEC, "Information Technology - Document Schema Definition Languages (DSDL) - Part 3: Rule-Based Validation - Schematron", ISO/IEC 19757-3:2006(E), June 2006.

- [XML] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <<http://www.w3.org/TR/2006/REC-xml-20060816>>.
- [XML-INFOSET] Tobin, R. and J. Cowan, "XML Information Set (Second Edition)", World Wide Web Consortium Recommendation REC-xml-infoset-20040204, February 2004, <<http://www.w3.org/TR/2004/REC-xml-infoset-20040204>>.
- [XPath] Clark, J. and S. DeRose, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <<http://www.w3.org/TR/1999/REC-xpath-19991116>>.
- [XSD-D] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-2-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.
- [XSLT] Clark, J., "XSL Transformations (XSLT) Version 1.0", World Wide Web Consortium Recommendation REC-xslt-19991116, November 1999.

## 17.2. Informative References

- [DHCPTut] Bjorklund, M., "DHCP Tutorial", November 2007, <<http://www.yang-central.org/twiki/bin/view/Main/DhcpTutorial>>.
- [RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", STD 15, RFC 1157, May 1990.
- [RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.
- [RFC5013] Kunze, J., "The Dublin Core Metadata Element Set", RFC 5013, August 2007.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.
- [Trang] Clark, J., "Trang: Multiformat schema converter based on RELAX NG", 2008, <<http://www.thaiopensource.com/relaxng/trang.html>>.

- [Vli04] van der Vlist, E., "RELAX NG", O'Reilly , 2004.
- [XSD] Thompson, H., Beech, D., Maloney, M., and N. Mendelsohn, "XML Schema Part 1: Structures Second Edition", World Wide Web Consortium Recommendation REC-xmlschema-1-20041028, October 2004, <<http://www.w3.org/TR/2004/REC-xmlschema-1-20041028>>.
- [pyang] Bjorklund, M. and L. Lhotka, "pyang: An extensible YANG validator and converter in Python", 2010, <<http://code.google.com/p/pyang/>>.



## Appendix A. RELAX NG Schema for NETMOD-Specific Annotations

This appendix defines the content model for all NETMOD-specific annotations in the form of RELAX NG named pattern definitions.

```
<CODE BEGINS> file "nmannot.rng"

<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:nma="urn:ietf:params:xml:ns:netmod:dSDL-annotations:1"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

  <define name="config-attribute">
    <attribute name="nma:config">
      <data type="boolean"/>
    </attribute>
  </define>

  <define name="data-element">
    <element name="nma:data">
      <ref name="__anyxml__"/>
    </element>
  </define>

  <define name="default-attribute">
    <attribute name="nma:default">
      <data type="string"/>
    </attribute>
  </define>

  <define name="error-app-tag-element">
    <element name="nma:error-app-tag">
      <text/>
    </element>
  </define>

  <define name="error-message-element">
    <element name="nma:error-message">
      <text/>
    </element>
  </define>

  <define name="if-feature-attribute">
    <attribute name="nma:if-feature">
      <list>
        <data type="QName"/>
      </list>
    </attribute>
  </define>
```

```
</define>

<define name="implicit-attribute">
  <attribute name="nma:implicit">
    <data type="boolean"/>
  </attribute>
</define>

<define name="instance-identifier-element">
  <element name="nma:instance-identifier">
    <optional>
      <attribute name="nma:require-instance">
        <data type="boolean"/>
      </attribute>
    </optional>
  </element>
</define>

<define name="key-attribute">
  <attribute name="nma:key">
    <list>
      <data type="QName"/>
    </list>
  </attribute>
</define>

<define name="leaf-list-attribute">
  <attribute name="nma:leaf-list">
    <data type="boolean"/>
  </attribute>
</define>

<define name="leafref-attribute">
  <attribute name="nma:leafref">
    <data type="string"/>
  </attribute>
</define>

<define name="mandatory-attribute">
  <attribute name="nma:mandatory">
    <data type="Name"/>
  </attribute>
</define>

<define name="max-elements-attribute">
  <attribute name="nma:max-elements">
    <data type="nonNegativeInteger"/>
  </attribute>
```

```
</define>

<define name="min-elements-attribute">
  <attribute name="nma:min-elements">
    <data type="nonNegativeInteger"/>
  </attribute>
</define>

<define name="module-attribute">
  <attribute name="nma:module">
    <data type="Name"/>
  </attribute>
</define>

<define name="must-element">
  <element name="nma:must">
    <attribute name="assert">
      <data type="string"/>
    </attribute>
    <interleave>
      <optional>
        <ref name="error-app-tag-element"/>
      </optional>
      <optional>
        <ref name="error-message-element"/>
      </optional>
    </interleave>
  </element>
</define>

<define name="notifications-element">
  <element name="nma:notifications">
    <zeroOrMore>
      <element name="nma:notification">
        <ref name="__anyxml__"/>
      </element>
    </zeroOrMore>
  </element>
</define>

<define name="rpcs-element">
  <element name="nma:rpcs">
    <zeroOrMore>
      <element name="nma:rpc">
        <interleave>
          <element name="nma:input">
            <ref name="__anyxml__"/>
          </element>
        </interleave>
      </element>
    </zeroOrMore>
  </element>
</define>
```

```
        <optional>
          <element name="nma:output">
            <ref name="__anyxml__"/>
          </element>
        </optional>
      </interleave>
    </element>
  </zeroOrMore>
</element>
</define>
```

```
<define name="ordered-by-attribute">
  <attribute name="nma:ordered-by">
    <choice>
      <value>user</value>
      <value>system</value>
    </choice>
  </attribute>
</define>
```

```
<define name="status-attribute">
  <optional>
    <attribute name="nma:status">
      <choice>
        <value>current</value>
        <value>deprecated</value>
        <value>obsolete</value>
      </choice>
    </attribute>
  </optional>
</define>
```

```
<define name="unique-attribute">
  <optional>
    <attribute name="nma:unique">
      <list>
        <data type="token"/>
      </list>
    </attribute>
  </optional>
</define>
```

```
<define name="units-attribute">
  <optional>
    <attribute name="nma:units">
      <data type="string"/>
    </attribute>
  </optional>
```

```
</define>

<define name="when-attribute">
  <optional>
    <attribute name="nma:when">
      <data type="string"/>
    </attribute>
  </optional>
</define>

<define name="__anyxml__">
  <zeroOrMore>
    <choice>
      <attribute>
        <anyName/>
      </attribute>
      <element>
        <anyName/>
        <ref name="__anyxml__"/>
      </element>
      <text/>
    </choice>
  </zeroOrMore>
</define>

</grammar>

<CODE ENDS>
```

## Appendix B. Schema-Independent Library

In order to avoid copying the common named pattern definitions to every RELAX NG schema generated in the second mapping step, the definitions are collected in a library file - schema-independent library - which is included by the validating schemas under the file name "relaxng-lib.rng" (XML syntax) and "relaxng-lib.rnc" (compact syntax). The included definitions cover patterns for common elements from base NETCONF [RFC4741] and event notifications [RFC5277].

```
<CODE BEGINS> file "relaxng-lib.rng"

<?xml version="1.0" encoding="UTF-8"?>

<grammar xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:en="urn:ietf:params:xml:ns:netconf:notification:1.0"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">

  <define name="message-id-attribute">
    <attribute name="message-id">
      <data type="string">
        <param name="maxLength">4095</param>
      </data>
    </attribute>
  </define>

  <define name="ok-element">
    <element name="nc:ok">
      <empty/>
    </element>
  </define>

  <define name="eventTime-element">
    <element name="en:eventTime">
      <data type="dateTime"/>
    </element>
  </define>
</grammar>

<CODE ENDS>
```

## Appendix C. Mapping DHCP Data Model - A Complete Example

This appendix demonstrates both steps of the YANG-to-DSDL mapping applied to the "canonical" DHCP tutorial [DHCPtut] data model. The input YANG module is shown in Appendix C.1 and the output schemas in the following two subsections.

The hybrid schema was obtained by the "dsdl" plugin of the pyang tool [pyang] and the validating DSDL schemas were obtained by XSLT stylesheets that are also part of pyang distribution.

Due to the limit of 72 characters per line, a few long strings required manual editing, in particular the regular expression patterns for IP addresses etc. These were replaced by the placeholder string "... regex pattern ...". Also, line breaks were added to several documentation strings and Schematron messages. Other than that, the results of the automatic translations were not changed.

## C.1. Input YANG Module

```
module dhcp {
  namespace "http://example.com/ns/dhcp";
  prefix dhcp;

  import ietf-yang-types { prefix yang; }
  import ietf-inet-types { prefix inet; }

  organization
    "yang-central.org";
  description
    "Partial data model for DHCP, based on the config of
    the ISC DHCP reference implementation.";

  container dhcp {
    description
      "configuration and operational parameters for a DHCP server.";

    leaf max-lease-time {
      type uint32;
      units seconds;
      default 7200;
    }

    leaf default-lease-time {
      type uint32;
      units seconds;
      must '.. <= ../max-lease-time' {
```

```
        error-message
          "The default-lease-time must be less than max-lease-time";
      }
      default 600;
    }

    uses subnet-list;

    container shared-networks {
      list shared-network {
        key name;

        leaf name {
          type string;
        }
        uses subnet-list;
      }
    }

    container status {
      config false;
      list leases {
        key address;

        leaf address {
          type inet:ip-address;
        }
        leaf starts {
          type yang:date-and-time;
        }
        leaf ends {
          type yang:date-and-time;
        }
        container hardware {
          leaf type {
            type enumeration {
              enum "ethernet";
              enum "token-ring";
              enum "fddi";
            }
          }
          leaf address {
            type yang:phys-address;
          }
        }
      }
    }
  }
}
```



```
grouping subnet-list {
  description "A reusable list of subnets";
  list subnet {
    key net;
    leaf net {
      type inet:ip-prefix;
    }
    container range {
      presence "enables dynamic address assignment";
      leaf dynamic-bootp {
        type empty;
        description
          "Allows BOOTP clients to get addresses in this range";
      }
      leaf low {
        type inet:ip-address;
        mandatory true;
      }
      leaf high {
        type inet:ip-address;
        mandatory true;
      }
    }
  }
}

container dhcp-options {
  description "Options in the DHCP protocol";
  leaf-list router {
    type inet:host;
    ordered-by user;
    reference "RFC 2132, sec. 3.8";
  }
  leaf domain-name {
    type inet:domain-name;
    reference "RFC 2132, sec. 3.17";
  }
}

leaf max-lease-time {
  type uint32;
  units seconds;
  default 7200;
}
}
```

## C.2. Hybrid Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:nma="urn:ietf:params:xml:ns:netmod:dSDL-annotations:1"
  xmlns:dc="http://purl.org/dc/terms"
  xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0"
  xmlns:dhcp="http://example.com/ns/dhcp"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <dc:creator>Pyang 1.0a, DSDL plugin</dc:creator>
  <dc:date>2010-06-17</dc:date>
  <start>
    <grammar nma:module="dhcp" ns="http://example.com/ns/dhcp">
      <dc:source>YANG module 'dhcp'</dc:source>
      <start>
        <nma:data>
          <optional>
            <element nma:implicit="true" name="dhcp:dhcp">
              <interleave>
                <a:documentation>
                  configuration and operational parameters for a DHCP server.
                </a:documentation>
              </interleave>
            </optional>
            <element nma:default="7200"
              name="dhcp:max-lease-time"
              nma:units="seconds">
              <data type="unsignedInt"/>
            </element>
          </optional>
          <optional>
            <element nma:default="600"
              name="dhcp:default-lease-time"
              nma:units="seconds">
              <data type="unsignedInt"/>
              <nma:must assert=". &lt;= ../dhcp:max-lease-time">
                <nma:error-message>
                  The default-lease-time must be less than max-lease-time
                </nma:error-message>
              </nma:must>
            </element>
          </optional>
          <ref name="_dhcp__subnet-list"/>
          <optional>
            <element name="dhcp:shared-networks">
              <zeroOrMore>
                <element nma:key="dhcp:name"
                  name="dhcp:shared-network">
                  <element name="dhcp:name">
                    <data type="string"/>
                  </element>
                </element>
              </zeroOrMore>
            </element>
          </optional>
        </nma:data>
      </start>
    </grammar>
  </start>
</grammar>
```

```
        </element>
        <ref name="_dhcp__subnet-list"/>
    </element>
</zeroOrMore>
</element>
</optional>
<optional>
    <element name="dhcp:status" nma:config="false">
        <zeroOrMore>
            <element nma:key="dhcp:address"
                name="dhcp:leases">
                <element name="dhcp:address">
                    <ref name="ietf-inet-types__ip-address"/>
                </element>
                <interleave>
                    <optional>
                        <element name="dhcp:starts">
                            <ref name="ietf-yang-types__date-and-time"/>
                        </element>
                    </optional>
                    <optional>
                        <element name="dhcp:ends">
                            <ref name="ietf-yang-types__date-and-time"/>
                        </element>
                    </optional>
                    <optional>
                        <element name="dhcp:hardware">
                            <interleave>
                                <optional>
                                    <element name="dhcp:type">
                                        <choice>
                                            <value>ethernet</value>
                                            <value>token-ring</value>
                                            <value>fddi</value>
                                        </choice>
                                    </element>
                                </optional>
                                <optional>
                                    <element name="dhcp:address">
                                        <ref name="ietf-yang-types__phys-address"/>
                                    </element>
                                </optional>
                            </interleave>
                        </element>
                    </optional>
                </interleave>
            </element>
        </zeroOrMore>
    </optional>
</optional>
```

```
        </element>
      </optional>
    </interleave>
  </element>
</optional>
</nma:data>
<nma:rpcs/>
<nma:notifications/>
</start>
</grammar>
</start>
<define name="ietf-yang-types__phys-address">
  <data type="string">
    <param name="pattern">([0-9a-fA-F]{2}(:[0-9a-fA-F]{2})*)?</param>
  </data>
</define>
<define name="ietf-inet-types__ipv6-address">
  <data type="string">
    <param name="pattern">... regex pattern ...</param>
    <param name="pattern">... regex pattern ...</param>
  </data>
</define>
<define name="ietf-inet-types__ip-prefix">
  <choice>
    <ref name="ietf-inet-types__ipv4-prefix"/>
    <ref name="ietf-inet-types__ipv6-prefix"/>
  </choice>
</define>
<define name="ietf-inet-types__host">
  <choice>
    <ref name="ietf-inet-types__ip-address"/>
    <ref name="ietf-inet-types__domain-name"/>
  </choice>
</define>
<define name="ietf-yang-types__date-and-time">
  <data type="string">
    <param name="pattern">... regex pattern ...</param>
  </data>
</define>
<define name="_dhcp__subnet-list">
  <a:documentation>A reusable list of subnets</a:documentation>
  <zeroOrMore>
    <element nma:key="net" name="subnet">
      <element name="net">
        <ref name="ietf-inet-types__ip-prefix"/>
      </element>
    </interleave>
  </optional>
```

```
<element name="range">
  <interleave>
    <optional>
      <element name="dynamic-bootp">
        <a:documentation>
          Allows BOOTP clients to get addresses in this range
        </a:documentation>
        <empty/>
      </element>
    </optional>
    <element name="low">
      <ref name="ietf-inet-types__ip-address"/>
    </element>
    <element name="high">
      <ref name="ietf-inet-types__ip-address"/>
    </element>
  </interleave>
</element>
</optional>
<optional>
  <element name="dhcp-options">
    <interleave>
      <a:documentation>
        Options in the DHCP protocol
      </a:documentation>
      <zeroOrMore>
        <element nma:leaf-list="true" name="router"
          nma:ordered-by="user">
          <a:documentation>
            See: RFC 2132, sec. 3.8
          </a:documentation>
          <ref name="ietf-inet-types__host"/>
        </element>
      </zeroOrMore>
      <optional>
        <element name="domain-name">
          <a:documentation>
            See: RFC 2132, sec. 3.17
          </a:documentation>
          <ref name="ietf-inet-types__domain-name"/>
        </element>
      </optional>
    </interleave>
  </element>
</optional>
<optional>
  <element nma:default="7200" name="max-lease-time"
    nma:units="seconds">
```

```
        <data type="unsignedInt"/>
      </element>
    </optional>
  </interleave>
</element>
</zeroOrMore>
</define>
<define name="ietf-inet-types__domain-name">
  <data type="string">
    <param name="pattern">... regex pattern ...</param>
    <param name="minLength">1</param>
    <param name="maxLength">253</param>
  </data>
</define>
<define name="ietf-inet-types__ipv4-prefix">
  <data type="string">
    <param name="pattern">... regex pattern ...</param>
  </data>
</define>
<define name="ietf-inet-types__ipv4-address">
  <data type="string">
    <param name="pattern">... regex pattern ...</param>
  </data>
</define>
<define name="ietf-inet-types__ipv6-prefix">
  <data type="string">
    <param name="pattern">... regex pattern ...</param>
    <param name="pattern">... regex pattern ...</param>
  </data>
</define>
<define name="ietf-inet-types__ip-address">
  <choice>
    <ref name="ietf-inet-types__ipv4-address"/>
    <ref name="ietf-inet-types__ipv6-address"/>
  </choice>
</define>
</grammar>
```

### C.3. Final DSDL Schemas

This appendix contains DSDL schemas that were obtained from the hybrid schema in Appendix C.2 by XSL transformations. These schemas can be directly used for validating a reply to unfiltered <nc:get> with the contents corresponding to the DHCP data model.

The RELAX NG schema (in two parts, as explained in Section 8.2) also includes the schema-independent library from Appendix B.

## C.3.1. Main RELAX NG Schema for &lt;nc:get&gt; Reply

```
<?xml version="1.0" encoding="utf-8"?>
<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:nma="urn:ietf:params:xml:ns:netmod:dSDL-annotations:1"
  xmlns:dhcp="http://example.com/ns/dhcp"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"
  ns="urn:ietf:params:xml:ns:netconf:base:1.0">
<include href="relaxng-lib.rng"/>
<start>
  <element name="rpc-reply">
    <ref name="message-id-attribute"/>
    <element name="data">
      <interleave>
        <grammar ns="http://example.com/ns/dhcp">
          <include href="dhcp-gdefs.rng"/>
          <start>
            <optional>
              <element name="dhcp:dhcp">
                <interleave>
                  <optional>
                    <element name="dhcp:max-lease-time">
                      <data type="unsignedInt"/>
                    </element>
                  </optional>
                  <optional>
                    <element name="dhcp:default-lease-time">
                      <data type="unsignedInt"/>
                    </element>
                  </optional>
                  <ref name="_dhcp__subnet-list"/>
                  <optional>
                    <element name="dhcp:shared-networks">
                      <zeroOrMore>
                        <element name="dhcp:shared-network">
                          <element name="dhcp:name">
                            <data type="string"/>
                          </element>
                          <ref name="_dhcp__subnet-list"/>
                        </element>
                      </zeroOrMore>
                    </element>
                  </optional>
                  <optional>
                    <element name="dhcp:status">
                      <zeroOrMore>
                        <element name="dhcp:leases">

```

```
<element name="dhcp:address">
  <ref name="ietf-inet-types__ip-address"/>
</element>
<interleave>
  <optional>
    <element name="dhcp:starts">
      <ref name="ietf-yang-types__date-and-time"/>
    </element>
  </optional>
  <optional>
    <element name="dhcp:ends">
      <ref name="ietf-yang-types__date-and-time"/>
    </element>
  </optional>
  <optional>
    <element name="dhcp:hardware">
      <interleave>
        <optional>
          <element name="dhcp:type">
            <choice>
              <value>ethernet</value>
              <value>token-ring</value>
              <value>fddi</value>
            </choice>
          </element>
        </optional>
        <optional>
          <element name="dhcp:address">
            <ref name="ietf-yang-types__phys-address"/>
          </element>
        </optional>
      </interleave>
    </element>
  </optional>
</interleave>
</element>
</zeroOrMore>
</element>
</optional>
</interleave>
</element>
</optional>
</start>
</grammar>
</interleave>
</element>
</element>
</start>
```



```
</grammar>
```

### C.3.2. RELAX NG Schema - Global Named Pattern Definitions

```
<?xml version="1.0" encoding="utf-8"?>
<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:nma="urn:ietf:params:xml:ns:netmod:dSDL-annotations:1"
  xmlns:dhcp="http://example.com/ns/dhcp"
  datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
  <define name="ietf-yang-types__phys-address">
    <data type="string">
      <param name="pattern">
        ([0-9a-fA-F]{2}(:[0-9a-fA-F]{2})*)?
      </param>
    </data>
  </define>
  <define name="ietf-inet-types__ipv6-address">
    <data type="string">
      <param name="pattern">... regex pattern ...</param>
    </data>
  </define>
  <define name="ietf-inet-types__ip-prefix">
    <choice>
      <ref name="ietf-inet-types__ipv4-prefix"/>
      <ref name="ietf-inet-types__ipv6-prefix"/>
    </choice>
  </define>
  <define name="ietf-inet-types__host">
    <choice>
      <ref name="ietf-inet-types__ip-address"/>
      <ref name="ietf-inet-types__domain-name"/>
    </choice>
  </define>
  <define name="ietf-yang-types__date-and-time">
    <data type="string">
      <param name="pattern">... regex pattern ...</param>
    </data>
  </define>
  <define name="__dhcp__subnet-list">
    <zeroOrMore>
      <element name="subnet">
        <element name="net">
          <ref name="ietf-inet-types__ip-prefix"/>
        </element>
        <interleave>
          <optional>
            <element name="range">
```

```
<interleave>
  <optional>
    <element name="dynamic-bootp">
      <empty/>
    </element>
  </optional>
  <element name="low">
    <ref name="ietf-inet-types__ip-address"/>
  </element>
  <element name="high">
    <ref name="ietf-inet-types__ip-address"/>
  </element>
</interleave>
</element>
</optional>
<optional>
  <element name="dhcp-options">
    <interleave>
      <zeroOrMore>
        <element name="router">
          <ref name="ietf-inet-types__host"/>
        </element>
      </zeroOrMore>
      <optional>
        <element name="domain-name">
          <ref name="ietf-inet-types__domain-name"/>
        </element>
      </optional>
    </interleave>
  </element>
</optional>
<optional>
  <element name="max-lease-time">
    <data type="unsignedInt"/>
  </element>
</optional>
</interleave>
</element>
</zeroOrMore>
</define>
<define name="ietf-inet-types__domain-name">
  <data type="string">
    <param name="pattern">... regex pattern ...</param>
    <param name="minLength">1</param>
    <param name="maxLength">253</param>
  </data>
</define>
<define name="ietf-inet-types__ipv4-prefix">
```

```

    <data type="string">
      <param name="pattern">... regex pattern ...</param>
    </data>
  </define>
  <define name="ietf-inet-types__ipv4-address">
    <data type="string">
      <param name="pattern">... regex pattern ...</param>
    </data>
  </define>
  <define name="ietf-inet-types__ipv6-prefix">
    <data type="string">
      <param name="pattern">... regex pattern ...</param>
      <param name="pattern">... regex pattern ...</param>
    </data>
  </define>
  <define name="ietf-inet-types__ip-address">
    <choice>
      <ref name="ietf-inet-types__ipv4-address"/>
      <ref name="ietf-inet-types__ipv6-address"/>
    </choice>
  </define>
</grammar>

```

### C.3.3. Schematron Schema for <nc:get> Reply

```

<?xml version="1.0" encoding="utf-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <sch:ns uri="http://example.com/ns/dhcp" prefix="dhcp"/>
  <sch:ns uri="urn:ietf:params:xml:ns:netconf:base:1.0" prefix="nc"/>
  <sch:pattern abstract="true" id="_dhcp__subnet-list">
    <sch:rule context="$start/$pref:subnet">
      <sch:report test="preceding-sibling::$pref:subnet
        [$pref:net=current()/$pref:net]">
        Duplicate key "net"
      </sch:report>
    </sch:rule>
    <sch:rule
      context="$start/$pref:subnet/$pref:dhcp-options/$pref:router">
      <sch:report test=".=preceding-sibling::router">
        Duplicate leaf-list value "<sch:value-of select="."/>"
      </sch:report>
    </sch:rule>
  </sch:pattern>
  <sch:pattern id="dhcp">
    <sch:rule
      context="/nc:rpc-reply/nc:data/dhcp:dhcp/dhcp:default-lease-time">
      <sch:assert test="<= ../dhcp:max-lease-time">
        The default-lease-time must be less than max-lease-time
    </sch:rule>
  </sch:pattern>

```

```
</sch:assert>
</sch:rule>
<sch:rule context="/nc:rpc-reply/nc:data/dhcp:dhcp/
                dhcp:shared-networks/dhcp:shared-network">
  <sch:report test="preceding-sibling::dhcp:shared-network
                  [dhcp:name=current()/dhcp:name]">
    Duplicate key "dhcp:name"
  </sch:report>
</sch:rule>
<sch:rule context="/nc:rpc-reply/nc:data/dhcp:dhcp/
                dhcp:status/dhcp:leases">
  <sch:report test="preceding-sibling::dhcp:leases
                  [dhcp:address=current()/dhcp:address]">
    Duplicate key "dhcp:address"
  </sch:report>
</sch:rule>
</sch:pattern>
<sch:pattern id="id2768196" is-a="_dhcp__subnet-list">
  <sch:param name="start" value="/nc:rpc-reply/nc:data/dhcp:dhcp"/>
  <sch:param name="pref" value="dhcp"/>
</sch:pattern>
<sch:pattern id="id2768215" is-a="_dhcp__subnet-list">
  <sch:param name="start"
              value="/nc:rpc-reply/nc:data/dhcp:dhcp/
                    dhcp:shared-networks/dhcp:shared-network"/>
  <sch:param name="pref" value="dhcp"/>
</sch:pattern>
</sch:schema>
```

## C.3.4. DSRL Schema for &lt;nc:get&gt; Reply

```
<?xml version="1.0" encoding="utf-8"?>
<dsrl:maps
  xmlns:dsrl="http://purl.oclc.org/dsdl/dsrl"
  xmlns:dhcp="http://example.com/ns/dhcp"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data</dsrl:parent>
    <dsrl:name>dhcp:dhcp</dsrl:name>
    <dsrl:default-content>
      <dhcp:max-lease-time>7200</dhcp:max-lease-time>
      <dhcp:default-lease-time>600</dhcp:default-lease-time>
    </dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data/dhcp:dhcp</dsrl:parent>
    <dsrl:name>dhcp:max-lease-time</dsrl:name>
    <dsrl:default-content>7200</dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>/nc:rpc-reply/nc:data/dhcp:dhcp</dsrl:parent>
    <dsrl:name>dhcp:default-lease-time</dsrl:name>
    <dsrl:default-content>600</dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>
      /nc:rpc-reply/nc:data/dhcp:dhcp/dhcp:subnet
    </dsrl:parent>
    <dsrl:name>dhcp:max-lease-time</dsrl:name>
    <dsrl:default-content>7200</dsrl:default-content>
  </dsrl:element-map>
  <dsrl:element-map>
    <dsrl:parent>
      /nc:rpc-reply/nc:data/dhcp:dhcp/dhcp:shared-networks/
      dhcp:shared-network/dhcp:subnet
    </dsrl:parent>
    <dsrl:name>dhcp:max-lease-time</dsrl:name>
    <dsrl:default-content>7200</dsrl:default-content>
  </dsrl:element-map>
</dsrl:maps>
```

## Appendix D. Change Log

RFC Editor: remove this section upon publication as an RFC.

## D.1. Changes Between Versions -07 and -08

- o Edits based on Gen-ART review.
- o Added formal templates in Section 13.
- o Created the "Contributors" section and moved the former co-authors there.
- o Indicated the location of both global and local named pattern definitions in the example hybrid schema in Section 8.1.
- o Added reference to EXSLT "evaluate" function.

## D.2. Changes Between Versions -06 and -07

- o Mapping of 'description', 'reference' and 'units' to the hybrid schema is now mandatory.
- o Improvements and fixes of the text based on the AD review

## D.3. Changes Between Versions -05 and -06

- o Terminology change: "conceptual tree schema" -> "hybrid schema".
- o Changed sectioning markers in the hybrid schema into plain NETMOD-specific annotations. Hence the former "nmt" namespace is not used at all.
- o Added the following NETMOD-specific annotations: @nma:if-feature, @nma:leaf-list, @nma:mandatory, @nma:module, removed @nma:presence.
- o Changed the structure of RELAX NG schemas by using embedded grammars and declaration of namespaces via @ns. This was necessary for enabling the "chameleon" behavior of global definitions.
- o Schematron validation phases are not used.
- o If an XPath expression appears inside a top-level grouping, the local prefix must be represented using the variable \$pref. (This is related to the previous item.)

- o DHCP example: All RNG schemas are only in the XML syntax. Added RNG with global definitions.
- o Added [XML-INFOSET] to normative references.
- o Listed the terms that are defined in other documents.
- o The schema for NETMOD-specific annotation is now given only as RNG named pattern definitions, no more in NVDL.

#### D.4. Changes Between Versions -04 and -05

- o Leafs that take their default value from a typedef and are not annotated with @nma:default must have @nma:implicit="true".
- o Changed code markers CODE BEGINS/ENDS to the form agreed by the WG.
- o Derived types "date-and-time" and "uri" SHOULD be mapped to XSD "dateTime" and "anyURI" types, respectively.
- o Clarified the notion of implicit nodes under 'case' in Section 9.1.2.
- o Moved draft-ietf-netmod-yang-types-06 to normative references.
- o An extra <rng:group> is no more required for the default case of a choice in the shorthand notation.

#### D.5. Changes Between Versions -03 and -04

- o Implemented ordering rules for list children - keys must go first and appear in the same order as in the input YANG module.
- o The 'case' statement is now mapped to either <rng:group> (inside RPC operations) or <rng:interleave> (otherwise).
- o A nma:default annotation coming from a datatype which the mapping expands is attached to the <rng:element> pattern where the expansion occurs. Added an example.
- o Documentation statements ('description', 'reference', 'status') MAY be ignored.
- o Single-valued numeric or length range parts are mapped to <rng:value> pattern or "length" facet.

- o Example for "string" datatype was added.
- o Appendix A now uses NVDL for defining NETMOD-specific annotations.
- o Added CODE BEGINS/ENDS markers.
- o Separated normative and informative references.
- o Added URL for XPath extensions namespace.
- o Added Section 2 (Terminology and Notation).
- o Added Section 14 (Security Considerations).
- o Added Section 16 (Acknowledgments).
- o Removed compact syntax schema from Appendix B.
- o Editorial changes: symbolic citation labels.

#### D.6. Changes Between Versions -02 and -03

- o Changed @nma:default-case to @nma:implicit.
- o Changed nma:leafref annotation from element to attribute.
- o Added skeleton rule to Section 11.2.
- o Reworked Section 11.3, added skeleton element maps, corrected the example.
- o Added section on 'feature' and 'deviation'.
- o New Section 9.1 integrating discussion of both optional/mandatory (was in -02) and implicit nodes (new).
- o Reflected that key argument and schema node identifiers are no more XPath (should be in yang-07).
- o Element patterns for implicit containers now must have @nma:implicit attribute.
- o Removed "float32" and "float64" types and added mapping of "decimal64" with example.
- o Removed mapping of 'require-instance' for "leafref" type.



- o Updated RELAX NG schema for NETMOD-specific annotations.
- o Updated the DHCP example.

#### D.7. Changes Between Versions -01 and -02

- o Moved Section 7 "NETCONF Content Validation" after Section 6.
- o New text about mapping defaults to DSRL, especially in Section 7 and Section 11.3.
- o Finished the DHCP example by adding the DSRL schema to Appendix C.
- o New @nma:presence annotation was added - it is needed for proper handling of default contents.
- o Section 11.2.1 "Constraints on Mandatory Choice" was added because these constraints require a combination of RELAX NG and Schematron.
- o Fixed the schema for NETMOD-specific annotations by adding explicit prefix to all defined elements and attributes. Previously, the attributes had no namespace.
- o Handling of 'feature', 'if-feature' and 'deviation' added.
- o Handling of nma:instance-identifier via XSLT extension function.

#### D.8. Changes Between Versions -00 and -01

- o Attributes @nma:min-elements and @nma:max-elements are attached to <rng:element> (list entry) and not to <rng:zeroOrMore> or <rng:oneOrMore>.
- o Keys and all node identifiers in 'key' and 'unique' statements are prefixed.
- o Fixed the mapping of 'rpc' and 'notification'.
- o Removed previous sec. 7.5 "RPC Signatures and Notifications" - the same information is now contained in Section 10.50 and Section 10.37.
- o Added initial "\_" to mangled names of groupings.
- o Mandated the use of @xmlns:xxx as the only method for declaring the target namespace.

- o Added section "Handling of XML Namespaces" to explain the previous item.
- o Completed DHCP example in Appendix C.
- o Almost all text about the second mapping step is new.

Author's Address

Ladislav Lhotka (editor)  
CESNET

Email: [lhotka@cesnet.cz](mailto:lhotka@cesnet.cz)



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: October 13, 2011

M. Bjorklund  
Tail-f Systems  
April 11, 2011

IANA Interface Type YANG Module  
draft-ietf-netmod-iana-if-type-00

Abstract

This document defines the initial version of the iana-if-type YANG module.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 13, 2011.

Copyright Notice

Copyright (c) 2011 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.

## Table of Contents

1. Introduction . . . . .	3
2. IANA Maintained Interface Type YANG Module . . . . .	4
3. IANA Considerations . . . . .	34
4. Security Considerations . . . . .	35
5. Normative References . . . . .	36
Author's Address . . . . .	37

## 1. Introduction

This document defines the initial version of the iana-if-type YANG module. This module reflects IANA's "ifType definitions" registry. The latest revision of the module can be obtained from the IANA web site.

## 2. IANA Maintained Interface Type YANG Module

```
<CODE BEGINS> file "iana-if-type.yang"

module iana-if-type {
  namespace "urn:ietf:params:xml:ns:yang:iana-if-type";
  prefix ift;

  organization "IANA";
  contact
    "      Internet Assigned Numbers Authority

    Postal: ICANN
           4676 Admiralty Way, Suite 330
           Marina del Rey, CA 90292

    Tel:    +1 310 823 9358
    E-Mail: iana@iana.org";
  description
    "This YANG module defines the iana-if-type typedef, which
    contains YANG definitions for IANA-registered interface types.

    The latest revision of this YANG module can be obtained from
    the IANA web site.

    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.: update the date below with the date of RFC publication
  // and remove this note.
  revision 2011-03-30 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: IANA Interface Type YANG Module";
  }
}
```



```
typedef iana-if-type {
  type enumeration {
    enum "other" {
      value 1;
      description
        "None of the following";
    }
    enum "regular1822" {
      value 2;
    }
    enum "hdl1822" {
      value 3;
    }
    enum "ddnX25" {
      value 4;
    }
    enum "rfc877x25" {
      value 5;
      reference
        "RFC 1382 - SNMP MIB Extension for the X.25 Packet Layer";
    }
    enum "ethernetCsmacd" {
      value 6;
      description
        "For all ethernet-like interfaces, regardless of speed,
         as per RFC3635.";
      reference
        "RFC 3635 - Definitions of Managed Objects for the
         Ethernet-like Interface Types.";
    }
    enum "iso88023Csmacd" {
      value 7;
      status deprecated;
      description
        "Deprecated via RFC3635.
         Use ethernetCsmacd(6) instead.";
      reference
        "RFC 3635 - Definitions of Managed Objects for the
         Ethernet-like Interface Types.";
    }
    enum "iso88024TokenBus" {
      value 8;
    }
    enum "iso88025TokenRing" {
      value 9;
    }
    enum "iso88026Man" {
      value 10;
    }
  }
}
```

```
}
enum "starLan" {
    value 11;
    status deprecated;
    description
        "Deprecated via RFC3635.
        Use ethernetCsmacd(6) instead.";
    reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types.";
}
enum "proteon10Mbit" {
    value 12;
}
enum "proteon80Mbit" {
    value 13;
}
enum "hyperchannel" {
    value 14;
}
enum "fddi" {
    value 15;
    reference
        "RFC 1512 - FDDI Management Information Base";
}
enum "lapb" {
    value 16;
    reference
        "RFC 1381 - SNMP MIB Extension for X.25 LAPB";
}
enum "sdlc" {
    value 17;
}
enum "dsl" {
    value 18;
    description
        "DSL-MIB";
    reference
        "RFC 4805 - Definitions of Managed Objects for the
        DSL, J1, E1, DS2, and E2 Interface Types";
}
enum "e1" {
    value 19;
    status obsolete;
    description
        "Obsolete see DSL-MIB";
    reference
        "RFC 4805 - Definitions of Managed Objects for the
```

```
        DS1, J1, E1, DS2, and E2 Interface Types";
    }
    enum "basicISDN" {
        value 20;
        description
            "see also RFC2127";
    }
    enum "primaryISDN" {
        value 21;
    }
    enum "propPointToPointSerial" {
        value 22;
        description
            "proprietary serial";
    }
    enum "ppp" {
        value 23;
    }
    enum "softwareLoopback" {
        value 24;
    }
    enum "eon" {
        value 25;
        description
            "CLNP over IP";
    }
    enum "ethernet3Mbit" {
        value 26;
    }
    enum "nsip" {
        value 27;
        description
            "XNS over IP";
    }
    enum "slip" {
        value 28;
        description
            "generic SLIP";
    }
    enum "ultra" {
        value 29;
        description
            "ULTRA technologies";
    }
    enum "ds3" {
        value 30;
        description
            "DS3-MIB";
    }
}
```

```
    reference
      "RFC 3896 - Definitions of Managed Objects for the
        DS3/E3 Interface Type";
  }
  enum "sip" {
    value 31;
    description
      "SMDS, coffee";
    reference
      "RFC 1694 - Definitions of Managed Objects for SMDS
        Interfaces using SMIV2";
  }
  enum "frameRelay" {
    value 32;
    description
      "DTE only.";
    reference
      "RFC 2115 - Management Information Base for Frame Relay
        DTEs Using SMIV2";
  }
  enum "rs232" {
    value 33;
    reference
      "RFC 1659 - Definitions of Managed Objects for RS-232-like
        Hardware Devices using SMIV2";
  }
  enum "para" {
    value 34;
    description
      "parallel-port";
    reference
      "RFC 1660 - Definitions of Managed Objects for
        Parallel-printer-like Hardware Devices using
        SMIV2";
  }
  enum "arcnet" {
    value 35;
    description
      "arcnet";
  }
  enum "arcnetPlus" {
    value 36;
    description
      "arcnet plus";
  }
  enum "atm" {
    value 37;
    description
```

```
        "ATM cells";
    }
    enum "miox25" {
        value 38;
        reference
            "RFC 1461 - SNMP MIB extension for Multiprotocol
              Interconnect over X.25";
    }
    enum "sonet" {
        value 39;
        description
            "SONET or SDH";
    }
    enum "x25ple" {
        value 40;
        reference
            "RFC 2127 - ISDN Management Information Base using SMIV2";
    }
    enum "iso88022llc" {
        value 41;
    }
    enum "localTalk" {
        value 42;
    }
    enum "smdsDxi" {
        value 43;
    }
    enum "frameRelayService" {
        value 44;
        description
            "FRNETSERV-MIB";
        reference
            "RFC 2954 - Definitions of Managed Objects for Frame
              Relay Service";
    }
    enum "v35" {
        value 45;
    }
    enum "hssi" {
        value 46;
    }
    enum "hippi" {
        value 47;
    }
    enum "modem" {
        value 48;
        description
            "Generic modem";
    }
}
```

```
}
enum "aal5" {
  value 49;
  description
    "AAL5 over ATM";
}
enum "sonetPath" {
  value 50;
}
enum "sonetVT" {
  value 51;
}
enum "smdsIcip" {
  value 52;
  description
    "SMDS InterCarrier Interface";
}
enum "propVirtual" {
  value 53;
  description
    "proprietary virtual/internal";
  reference
    "RFC 2863 - The Interfaces Group MIB";
}
enum "propMultiplexor" {
  value 54;
  description
    "proprietary multiplexing";
  reference
    "RFC 2863 - The Interfaces Group MIB";
}
enum "ieee80212" {
  value 55;
  description
    "100BaseVG";
}
enum "fibreChannel" {
  value 56;
  description
    "Fibre Channel";
}
enum "hippiInterface" {
  value 57;
  description
    "HIPPI interfaces";
}
enum "frameRelayInterconnect" {
  value 58;
```

```
    status obsolete;
    description
        "Obsolete use either
         frameRelay(32) or frameRelayService(44).";
}
enum "aflane8023" {
    value 59;
    description
        "ATM Emulated LAN for 802.3";
}
enum "aflane8025" {
    value 60;
    description
        "ATM Emulated LAN for 802.5";
}
enum "cctEmul" {
    value 61;
    description
        "ATM Emulated circuit";
}
enum "fastEther" {
    value 62;
    status deprecated;
    description
        "Obsoleted via RFC3635.
         ethernetCsmacd(6) should be used instead";
    reference
        "RFC 3635 - Definitions of Managed Objects for the
         Ethernet-like Interface Types.";
}
enum "isdn" {
    value 63;
    description
        "ISDN and X.25";
    reference
        "RFC 1356 - Multiprotocol Interconnect on X.25 and ISDN
         in the Packet Mode";
}
enum "v11" {
    value 64;
    description
        "CCITT V.11/X.21";
}
enum "v36" {
    value 65;
    description
        "CCITT V.36";
}
```

```
enum "g703at64k" {
    value 66;
    description
        "CCITT G703 at 64Kbps";
}
enum "g703at2mb" {
    value 67;
    status obsolete;
    description
        "Obsolete see DS1-MIB";
}
enum "qllc" {
    value 68;
    description
        "SNA QLLC";
}
enum "fastEtherFX" {
    value 69;
    status deprecated;
    description
        "Obsoleted via RFC3635
        ethernetCsmacd(6) should be used instead";
    reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types.";
}
enum "channel" {
    value 70;
    description
        "channel";
}
enum "ieee80211" {
    value 71;
    description
        "radio spread spectrum";
}
enum "ibm370parChan" {
    value 72;
    description
        "IBM System 360/370 OEMI Channel";
}
enum "escon" {
    value 73;
    description
        "IBM Enterprise Systems Connection";
}
enum "dlsu" {
    value 74;
```



```
        description
            "Data Link Switching";
    }
    enum "isdns" {
        value 75;
        description
            "ISDN S/T interface";
    }
    enum "isdnu" {
        value 76;
        description
            "ISDN U interface";
    }
    enum "lapd" {
        value 77;
        description
            "Link Access Protocol D";
    }
    enum "ipSwitch" {
        value 78;
        description
            "IP Switching Objects";
    }
    enum "rsrb" {
        value 79;
        description
            "Remote Source Route Bridging";
    }
    enum "atmLogical" {
        value 80;
        description
            "ATM Logical Port";
        reference
            "RFC 3606 - Definitions of Supplemental Managed Objects
            for ATM Interface";
    }
    enum "ds0" {
        value 81;
        description
            "Digital Signal Level 0";
        reference
            "RFC 2494 - Definitions of Managed Objects for the DS0
            and DS0 Bundle Interface Type";
    }
    enum "ds0Bundle" {
        value 82;
        description
            "group of ds0s on the same ds1";
    }
```

```
    reference
      "RFC 2494 - Definitions of Managed Objects for the DS0
        and DS0 Bundle Interface Type";
  }
  enum "bsc" {
    value 83;
    description
      "Bisynchronous Protocol";
  }
  enum "async" {
    value 84;
    description
      "Asynchronous Protocol";
  }
  enum "cnr" {
    value 85;
    description
      "Combat Net Radio";
  }
  enum "iso88025Dtr" {
    value 86;
    description
      "ISO 802.5r DTR";
  }
  enum "eplrs" {
    value 87;
    description
      "Ext Pos Loc Report Sys";
  }
  enum "arap" {
    value 88;
    description
      "Appletalk Remote Access Protocol";
  }
  enum "propCnls" {
    value 89;
    description
      "Proprietary Connectionless Protocol";
  }
  enum "hostPad" {
    value 90;
    description
      "CCITT-ITU X.29 PAD Protocol";
  }
  enum "termPad" {
    value 91;
    description
      "CCITT-ITU X.3 PAD Facility";
  }
```

```
}
enum "frameRelayMPI" {
    value 92;
    description
        "Multiproto Interconnect over FR";
}
enum "x213" {
    value 93;
    description
        "CCITT-ITU X213";
}
enum "adsl" {
    value 94;
    description
        "Asymmetric Digital Subscriber Loop";
}
enum "radsl" {
    value 95;
    description
        "Rate-Adapt. Digital Subscriber Loop";
}
enum "sdsl" {
    value 96;
    description
        "Symmetric Digital Subscriber Loop";
}
enum "vdsl" {
    value 97;
    description
        "Very H-Speed Digital Subscrib. Loop";
}
enum "iso88025CRFPInt" {
    value 98;
    description
        "ISO 802.5 CRFP";
}
enum "myrinet" {
    value 99;
    description
        "Myricom Myrinet";
}
enum "voiceEM" {
    value 100;
    description
        "voice receive and transmit";
}
enum "voiceFXO" {
    value 101;
```

```
    description
      "voice Foreign Exchange Office";
  }
  enum "voiceFXS" {
    value 102;
    description
      "voice Foreign Exchange Station";
  }
  enum "voiceEncap" {
    value 103;
    description
      "voice encapsulation";
  }
  enum "voiceOverIp" {
    value 104;
    description
      "voice over IP encapsulation";
  }
  enum "atmDxi" {
    value 105;
    description
      "ATM DXI";
  }
  enum "atmFuni" {
    value 106;
    description
      "ATM FUNI";
  }
  enum "atmIma" {
    value 107;
    description
      "ATM IMA";
  }
  enum "pppMultilinkBundle" {
    value 108;
    description
      "PPP Multilink Bundle";
  }
  enum "ipOverCdlc" {
    value 109;
    description
      "IBM ipOverCdlc";
  }
  enum "ipOverClaw" {
    value 110;
    description
      "IBM Common Link Access to Workstn";
  }
}
```

```
enum "stackToStack" {
    value 111;
    description
        "IBM stackToStack";
}
enum "virtualIpAddress" {
    value 112;
    description
        "IBM VIPA";
}
enum "mpc" {
    value 113;
    description
        "IBM multi-protocol channel support";
}
enum "ipOverAtm" {
    value 114;
    description
        "IBM ipOverAtm";
    reference
        "RFC 2320 - Definitions of Managed Objects for Classical IP
        and ARP Over ATM Using SMIv2 (IPOA-MIB)";
}
enum "iso88025Fiber" {
    value 115;
    description
        "ISO 802.5j Fiber Token Ring";
}
enum "tdlc" {
    value 116;
    description
        "IBM twinaxial data link control";
}
enum "gigabitEthernet" {
    value 117;
    status deprecated;
    description
        "Obsoleted via RFC3635
        ethernetCsmacd(6) should be used instead";
    reference
        "RFC 3635 - Definitions of Managed Objects for the
        Ethernet-like Interface Types.";
}
enum "hdlc" {
    value 118;
    description
        "HDLC";
}
```

```
enum "lapf" {  
    value 119;  
    description  
        "LAP F";  
}  
enum "v37" {  
    value 120;  
    description  
        "V.37";  
}  
enum "x25mlp" {  
    value 121;  
    description  
        "Multi-Link Protocol";  
}  
enum "x25huntGroup" {  
    value 122;  
    description  
        "X25 Hunt Group";  
}  
enum "transpHdlc" {  
    value 123;  
    description  
        "Transp HDLC";  
}  
enum "interleave" {  
    value 124;  
    description  
        "Interleave channel";  
}  
enum "fast" {  
    value 125;  
    description  
        "Fast channel";  
}  
enum "ip" {  
    value 126;  
    description  
        "IP (for APPN HPR in IP networks)";  
}  
enum "docsCableMacLayer" {  
    value 127;  
    description  
        "CATV Mac Layer";  
}  
enum "docsCableDownstream" {  
    value 128;  
    description
```

```
        "CATV Downstream interface";
    }
    enum "docsCableUpstream" {
        value 129;
        description
            "CATV Upstream interface";
    }
    enum "a12MppSwitch" {
        value 130;
        description
            "Avalon Parallel Processor";
    }
    enum "tunnel" {
        value 131;
        description
            "Encapsulation interface";
    }
    enum "coffee" {
        value 132;
        description
            "coffee pot";
        reference
            "RFC 2325 - Coffee MIB";
    }
    enum "ces" {
        value 133;
        description
            "Circuit Emulation Service";
    }
    enum "atmSubInterface" {
        value 134;
        description
            "ATM Sub Interface";
    }
    enum "l2vlan" {
        value 135;
        description
            "Layer 2 Virtual LAN using 802.1Q";
    }
    enum "l3ipvlan" {
        value 136;
        description
            "Layer 3 Virtual LAN using IP";
    }
    enum "l3ipxvlan" {
        value 137;
        description
            "Layer 3 Virtual LAN using IPX";
    }
```

```
}
enum "digitalPowerline" {
    value 138;
    description
        "IP over Power Lines";
}
enum "mediaMailOverIp" {
    value 139;
    description
        "Multimedia Mail over IP";
}
enum "dtm" {
    value 140;
    description
        "Dynamic synchronous Transfer Mode";
}
enum "dcn" {
    value 141;
    description
        "Data Communications Network";
}
enum "ipForward" {
    value 142;
    description
        "IP Forwarding Interface";
}
enum "msdsl" {
    value 143;
    description
        "Multi-rate Symmetric DSL";
}
enum "ieee1394" {
    value 144;
    description
        "IEEE1394 High Performance Serial Bus";
}
enum "if-gsn" {
    value 145;
    description
        "HIPPI-6400";
}
enum "dvbRccMacLayer" {
    value 146;
    description
        "DVB-RCC MAC Layer";
}
enum "dvbRccDownstream" {
    value 147;
```



```
        description
        "DVB-RCC Downstream Channel";
    }
    enum "dvbRccUpstream" {
        value 148;
        description
        "DVB-RCC Upstream Channel";
    }
    enum "atmVirtual" {
        value 149;
        description
        "ATM Virtual Interface";
    }
    enum "mplsTunnel" {
        value 150;
        description
        "MPLS Tunnel Virtual Interface";
    }
    enum "srp" {
        value 151;
        description
        "Spatial Reuse Protocol          ";
    }
    enum "voiceOverAtm" {
        value 152;
        description
        "Voice Over ATM";
    }
    enum "voiceOverFrameRelay" {
        value 153;
        description
        "Voice Over Frame Relay";
    }
    enum "ids1" {
        value 154;
        description
        "Digital Subscriber Loop over ISDN";
    }
    enum "compositeLink" {
        value 155;
        description
        "Avici Composite Link Interface";
    }
    enum "ss7SigLink" {
        value 156;
        description
        "SS7 Signaling Link";
    }
}
```

```
enum "propWirelessP2P" {
    value 157;
    description
        "Prop. P2P wireless interface";
}
enum "frForward" {
    value 158;
    description
        "Frame Forward Interface";
}
enum "rfc1483" {
    value 159;
    description
        "Multiprotocol over ATM AAL5";
    reference
        "RFC 1483 - Multiprotocol Encapsulation over ATM
        Adaptation Layer 5";
}
enum "usb" {
    value 160;
    description
        "USB Interface";
}
enum "ieee8023adLag" {
    value 161;
    description
        "IEEE 802.3ad Link Aggregate";
}
enum "bgppolicyaccounting" {
    value 162;
    description
        "BGP Policy Accounting";
}
enum "frfl6MfrBundle" {
    value 163;
    description
        "FRF .16 Multilink Frame Relay";
}
enum "h323Gatekeeper" {
    value 164;
    description
        "H323 Gatekeeper";
}
enum "h323Proxy" {
    value 165;
    description
        "H323 Voice and Video Proxy";
}
```

```
enum "mpls" {  
    value 166;  
    description  
        "MPLS";  
}  
enum "mfSigLink" {  
    value 167;  
    description  
        "Multi-frequency signaling link";  
}  
enum "hdsl2" {  
    value 168;  
    description  
        "High Bit-Rate DSL - 2nd generation";  
}  
enum "shdsl" {  
    value 169;  
    description  
        "Multirate HDSL2";  
}  
enum "dslFDL" {  
    value 170;  
    description  
        "Facility Data Link 4Kbps on a DS1";  
}  
enum "pos" {  
    value 171;  
    description  
        "Packet over SONET/SDH Interface";  
}  
enum "dvbAsiIn" {  
    value 172;  
    description  
        "DVB-ASI Input";  
}  
enum "dvbAsiOut" {  
    value 173;  
    description  
        "DVB-ASI Output";  
}  
enum "plc" {  
    value 174;  
    description  
        "Power Line Communtications";  
}  
enum "nfas" {  
    value 175;  
    description
```

```
        "Non Facility Associated Signaling";
    }
    enum "tr008" {
        value 176;
        description
            "TR008";
    }
    enum "gr303RDT" {
        value 177;
        description
            "Remote Digital Terminal";
    }
    enum "gr303IDT" {
        value 178;
        description
            "Integrated Digital Terminal";
    }
    enum "isup" {
        value 179;
        description
            "ISUP";
    }
    enum "propDocsWirelessMaclayer" {
        value 180;
        description
            "Cisco proprietary Maclayer";
    }
    enum "propDocsWirelessDownstream" {
        value 181;
        description
            "Cisco proprietary Downstream";
    }
    enum "propDocsWirelessUpstream" {
        value 182;
        description
            "Cisco proprietary Upstream";
    }
    enum "hiperlan2" {
        value 183;
        description
            "HIPERLAN Type 2 Radio Interface";
    }
    enum "propBWAp2Mp" {
        value 184;
        description
            "PropBroadbandWirelessAccesspt2multipt use of this value
            for IEEE 802.16 WMAN interfaces as per IEEE Std 802.16f
            is deprecated and ieee80216WMAN(237) should be used
    }
```

```
        instead.";
    }
    enum "sonetOverheadChannel" {
        value 185;
        description
            "SONET Overhead Channel";
    }
    enum "digitalWrapperOverheadChannel" {
        value 186;
        description
            "Digital Wrapper";
    }
    enum "aal2" {
        value 187;
        description
            "ATM adaptation layer 2";
    }
    enum "radioMAC" {
        value 188;
        description
            "MAC layer over radio links";
    }
    enum "atmRadio" {
        value 189;
        description
            "ATM over radio links";
    }
    enum "imt" {
        value 190;
        description
            "Inter Machine Trunks";
    }
    enum "mv1" {
        value 191;
        description
            "Multiple Virtual Lines DSL";
    }
    enum "reachDSL" {
        value 192;
        description
            "Long Reach DSL";
    }
    enum "frDlciEndPt" {
        value 193;
        description
            "Frame Relay DLCI End Point";
    }
    enum "atmVciEndPt" {
```

```
        value 194;
        description
            "ATM VCI End Point";
    }
    enum "opticalChannel" {
        value 195;
        description
            "Optical Channel";
    }
    enum "opticalTransport" {
        value 196;
        description
            "Optical Transport";
    }
    enum "propAtm" {
        value 197;
        description
            "Proprietary ATM";
    }
    enum "voiceOverCable" {
        value 198;
        description
            "Voice Over Cable Interface";
    }
    enum "infiniband" {
        value 199;
        description
            "Infiniband";
    }
    enum "teLink" {
        value 200;
        description
            "TE Link";
    }
    enum "q2931" {
        value 201;
        description
            "Q.2931";
    }
    enum "virtualTg" {
        value 202;
        description
            "Virtual Trunk Group";
    }
    enum "sipTg" {
        value 203;
        description
            "SIP Trunk Group";
    }
```

```
}
enum "sipSig" {
  value 204;
  description
    "SIP Signaling";
}
enum "docsCableUpstreamChannel" {
  value 205;
  description
    "CATV Upstream Channel";
}
enum "econet" {
  value 206;
  description
    "Acorn Econet";
}
enum "pon155" {
  value 207;
  description
    "FSAN 155Mb Symmetrical PON interface";
}
enum "pon622" {
  value 208;
  description
    "FSAN622Mb Symmetrical PON interface";
}
enum "bridge" {
  value 209;
  description
    "Transparent bridge interface";
}
enum "linegroup" {
  value 210;
  description
    "Interface common to multiple lines";
}
enum "voiceEMFGD" {
  value 211;
  description
    "voice E&M Feature Group D";
}
enum "voiceFGDEANA" {
  value 212;
  description
    "voice FGD Exchange Access North American";
}
enum "voiceDID" {
  value 213;
```

```
    description
      "voice Direct Inward Dialing";
  }
  enum "mpegTransport" {
    value 214;
    description
      "MPEG transport interface";
  }
  enum "sixToFour" {
    value 215;
    status deprecated;
    description
      "6to4 interface (DEPRECATED)";
    reference
      "RFC 4087 - IP Tunnel MIB";
  }
  enum "gtp" {
    value 216;
    description
      "GTP (GPRS Tunneling Protocol)";
  }
  enum "pdnEtherLoop1" {
    value 217;
    description
      "Paradyne EtherLoop 1";
  }
  enum "pdnEtherLoop2" {
    value 218;
    description
      "Paradyne EtherLoop 2";
  }
  enum "opticalChannelGroup" {
    value 219;
    description
      "Optical Channel Group";
  }
  enum "homepna" {
    value 220;
    description
      "HomePNA ITU-T G.989";
  }
  enum "gfp" {
    value 221;
    description
      "Generic Framing Procedure (GFP)";
  }
  enum "ciscoISLvlan" {
    value 222;
```



```
    description
      "Layer 2 Virtual LAN using Cisco ISL";
  }
  enum "actelisMetaLOOP" {
    value 223;
    description
      "Acteleis proprietary MetaLOOP High Speed Link";
  }
  enum "fcipLink" {
    value 224;
    description
      "FCIP Link";
  }
  enum "rpr" {
    value 225;
    description
      "Resilient Packet Ring Interface Type";
  }
  enum "qam" {
    value 226;
    description
      "RF Qam Interface";
  }
  enum "lmp" {
    value 227;
    description
      "Link Management Protocol";
    reference
      "RFC 4327 - Link Management Protocol (LMP) Management
        Information Base (MIB)";
  }
  enum "cblVectaStar" {
    value 228;
    description
      "Cambridge Broadband Networks Limited VectaStar";
  }
  enum "docsCableMCmtsDownstream" {
    value 229;
    description
      "CATV Modular CMTS Downstream Interface";
  }
  enum "adsl2" {
    value 230;
    status deprecated;
    description
      "Asymmetric Digital Subscriber Loop Version 2
        (DEPRECATED/OBSOLETE - please use adsl2plus(238)
        instead)";
  }
```

```
    reference
      "RFC 4706 - Definitions of Managed Objects for Asymmetric
        Digital Subscriber Line 2 (ADSL2)";
  }
  enum "macSecControlledIF" {
    value 231;
    description
      "MACSecControlled";
  }
  enum "macSecUncontrolledIF" {
    value 232;
    description
      "MACSecUncontrolled";
  }
  enum "aviciOpticalEther" {
    value 233;
    description
      "Avici Optical Ethernet Aggregate";
  }
  enum "atmbond" {
    value 234;
    description
      "atmbond";
  }
  enum "voiceFGDOS" {
    value 235;
    description
      "voice FGD Operator Services";
  }
  enum "mocaVersion1" {
    value 236;
    description
      "MultiMedia over Coax Alliance (MoCA) Interface
        as documented in information provided privately to IANA";
  }
  enum "ieee80216WMAN" {
    value 237;
    description
      "IEEE 802.16 WMAN interface";
  }
  enum "adsl2plus" {
    value 238;
    description
      "Asymmetric Digital Subscriber Loop Version 2,
        Version 2 Plus and all variants";
  }
  enum "dvbRcsMacLayer" {
    value 239;
```

```
    description
        "DVB-RCS MAC Layer";
    reference
        "RFC 5728 - The SatLabs Group DVB-RCS MIB";
}
enum "dvbTdm" {
    value 240;
    description
        "DVB Satellite TDM";
    reference
        "RFC 5728 - The SatLabs Group DVB-RCS MIB";
}
enum "dvbRcsTdma" {
    value 241;
    description
        "DVB-RCS TDMA";
    reference
        "RFC 5728 - The SatLabs Group DVB-RCS MIB";
}
enum "x86Laps" {
    value 242;
    description
        "LAPS based on ITU-T X.86/Y.1323";
}
enum "wwanPP" {
    value 243;
    description
        "3GPP WWAN";
}
enum "wwanPP2" {
    value 244;
    description
        "3GPP2 WWAN";
}
enum "voiceEBS" {
    value 245;
    description
        "voice P-phone EBS physical interface";
}
enum "ifPwType" {
    value 246;
    description
        "Pseudowire interface type";
    reference
        "RFC 5601 - Pseudowire (PW) Management Information Base";
}
enum "ilan" {
    value 247;
```

```
    description
      "Internal LAN on a bridge per IEEE 802.1ap";
  }
  enum "pip" {
    value 248;
    description
      "Provider Instance Port on a bridge per IEEE 802.1ah PBB";
  }
  enum "aluELP" {
    value 249;
    description
      "Alcatel-Lucent Ethernet Link Protection";
  }
  enum "gpon" {
    value 250;
    description
      "Gigabit-capable passive optical networks (G-PON) as per
      ITU-T G.948";
  }
  enum "vdsl2" {
    value 251;
    description
      "Very high speed digital subscriber line Version 2
      (as per ITU-T Recommendation G.993.2)";
    reference
      "RFC 5650 - Definitions of Managed Objects for Very High
      Speed Digital Subscriber Line 2 (VDSL2)";
  }
  enum "capwapDot11Profile" {
    value 252;
    description
      "WLAN Profile Interface";
    reference
      "RFC 5834 - Control and Provisioning of Wireless Access
      Points (CAPWAP) Protocol Binding MIB for
      IEEE 802.11";
  }
  enum "capwapDot11Bss" {
    value 253;
    description
      "WLAN BSS Interface";
    reference
      "RFC 5834 - Control and Provisioning of Wireless Access
      Points (CAPWAP) Protocol Binding MIB for
      IEEE 802.11";
  }
  enum "capwapWtpVirtualRadio" {
    value 254;
```

```
        description
            "WTP Virtual Radio Interface";
        reference
            "RFC 5833 - Control and Provisioning of Wireless Access
              Points (CAPWAP) Protocol Base MIB";
    }
    enum "bits" {
        value 255;
        description
            "bitsport";
    }
    enum "docsCableUpstreamRfPort" {
        value 256;
        description
            "DOCSIS CATV Upstream RF Port";
    }
    enum "cableDownstreamRfPort" {
        value 257;
        description
            "CATV downstream RF port";
    }
}
description
    "This data type is used as the syntax of the 'type'
    leaf in the 'interface' list in the YANG module
    ietf-interface.

    The definition of this typedef with the
    addition of newly assigned values is published
    periodically by the IANA, in either the Assigned
    Numbers RFC, or some derivative of it specific to
    Internet Network Management number assignments. (The
    latest arrangements can be obtained by contacting the
    IANA.)

    Requests for new values should be made to IANA via
    email (iana@iana.org).";
}
}
```

<CODE ENDS>

### 3. IANA Considerations

This document defines the initial version of the IANA-maintained iana-if-type YANG module. The iana-if-type module is intended to reflect the "ifType definitions" registry. When an interface type is added to this registry, a new "enum" statement must be added to the "iana-if-type" typedef, with the same name and value as the corresponding enumeration in IANAifType-MIB. If the new interface type has a reference, a new "reference" statement should be added to the new "enum" statement. If an interface type is deprecated in the "ifType definitions" registry, the corresponding "enum" statement must be updated with a "status" statement with the value "deprecated".

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

URI: urn:ietf:params:xml:ns:yang:iana-if-types

Registrant Contact: The NETMOD WG of the IETF.

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

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

name:	iana-if-type
namespace:	urn:ietf:params:xml:ns:yang:iana-if-type
prefix:	ift
reference:	RFC XXXX

#### 4. Security Considerations

Since this document does not introduce any technology or protocol, there are no security issues to be considered for this document itself.

## 5. Normative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.



Author's Address

Martin Bjorklund  
Tail-f Systems

Email: [mbj@tail-f.com](mailto:mbj@tail-f.com)



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: November 21, 2011

M. Bjorklund  
Tail-f Systems  
May 20, 2011

A YANG Data Model for Interface Configuration  
draft-ietf-netmod-interfaces-cfg-01

Abstract

This document defines a YANG data model for the configuration of network interfaces. It is expected that interface type specific configuration data models augment the generic interfaces data model defined in this document.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 21, 2011.

Copyright Notice

Copyright (c) 2011 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.

## Table of Contents

1. Introduction . . . . .	3
2. Objectives . . . . .	4
3. Interfaces Data Model . . . . .	5
3.1. The interface List . . . . .	5
3.2. Interface References . . . . .	6
3.3. Interface Layering . . . . .	6
4. Interfaces YANG Module . . . . .	7
5. IANA Considerations . . . . .	12
6. Security Considerations . . . . .	13
7. Acknowledgments . . . . .	14
8. Normative References . . . . .	15
Appendix A. Example: Ethernet Interface Module . . . . .	16
Appendix B. Example: Ethernet Bonding Interface Module . . . . .	18
Appendix C. Example: VLAN Interface Module . . . . .	19
Appendix D. Example: IP Module . . . . .	20
Appendix E. Example: NETCONF <get> reply . . . . .	21
Appendix F. ChangeLog . . . . .	22
F.1. Version -01 . . . . .	22
Author's Address . . . . .	23

## 1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of network interfaces. It is expected that interface type specific configuration data models augment the generic interfaces data model defined in this document.

Network interfaces are central to the configuration of many Internet protocols. Thus, it is important to establish a common data model for how interfaces are identified and configured.

The keywords "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. Objectives

This section describes some of the design objectives for the model presented in Section 4.

- o It is recognized that existing implementations will have to map the interface data model defined in this memo to their proprietary native data model. The new data model should be simple to facilitate such mappings.
- o The data model should be suitable for new implementations to use as-is, without requiring a mapping to a different native model.
- o The data model must be extensible for different specific interface types, including vendor-specific types.
- o References to interfaces should be as simple as possible, preferably by using a single leafref.
- o The mapping to ifIndex [RFC2863] used by SNMP to identify interfaces must be clear.
- o The model must support interface layering, both simple layering where one interface is layered on top of exactly one other interface, and more complex scenarios where one interface is aggregated over N other interfaces, or when N interfaces are multiplexed over one other interface.
- o The data model should support the pre-provisioning of interface configuration, i.e., it should be possible to configure an interface whose physical interface hardware is not present on the device. It is recommended that devices that support dynamic addition and removal of physical interfaces also support pre-provisioning.

### 3. Interfaces Data Model

The module "ietf-interfaces" has the following structure:

```

+--rw interfaces
  +--rw interface [name]
    +--rw name                string
    +--rw description?        string
    +--rw type                 ift:iana-if-type
    +--rw location?           string
    +--rw enabled?            boolean
    +--ro if-index*           int32
    +--rw mtu?                uint32
    +--rw link-up-down-trap-enable? enumeration

```

This module defines one YANG feature:

snmp-if-mib: Indicates that the server implements IF-MIB [RFC2863].

#### 3.1. The interface List

The data model for interface configuration presented in this document uses a flat list of interfaces. Each interface in the list is identified by its name. Furthermore, each interface has a mandatory "type" leaf, and a "location" leaf. The combination of "type" and "location" is unique within the interface list.

It is expected that interface type specific data models augment the interface list, and use the "type" leaf to make the augmentation conditional.

As an example of such an interface type specific augmentation, consider this YANG snippet. For a more complete example, see Appendix A.

```

import interfaces {
  prefix "if";
}

augment "/if:interfaces/if:interface" {
  when "if:type = 'ethernetCsmacd'";

  container ethernet {
    leaf duplex {
      ...
    }
  }
}

```

The "location" leaf is a string. It is optional in the data model, but if the type represents a physical interface, it is mandatory. The format of this string is device- and type-dependent. The device uses the location string to identify the physical or logical entity that the configuration applies to. For example, if a device has a single array of 8 ethernet ports, the location can be one of the strings "1" to "8". As another example, if a device has N cards of M ports, the location can be on the form "n/m", such as "1/0".

How a client can learn which types and locations are present on a certain device is outside the scope of this document.

### 3.2. Interface References

An interface is uniquely identified by its name. This property is captured in the "interface-ref" typedef, which other YANG modules SHOULD use when they need to reference an existing interface.

### 3.3. Interface Layering

There is no generic mechanism for how an interface is configured to be layered on top of some other interface. It is expected that interface type specific models define their own objects for interface layering, by using "interface-ref" types to reference lower layers.

Below is an example of a model with such objects. For a more complete example, see Appendix B.

```
augment "/if:interfaces/if:interface" {
  when "if:type = 'ieee8023adLag'";

  leaf-list slave-if {
    type if:interface-ref;
    must "/if:interfaces/if:interface[if:name = current()]"
      + "/if:type = 'eth:ethernet'" {
      description
        "The type of a slave interface must be ethernet";
    }
  }
  // other bonding config params, failover times etc.
}
```



#### 4. Interfaces YANG Module

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-interfaces@2011-05-20.yang"

```
module ietf-interfaces {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-interfaces";  
    prefix if;  
  
    import iana-if-type {  
        prefix ift;  
    }  
  
    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:     Martin Bjorklund  
                   <mailto:mbj@tail-f.com>";  
  
    description  
        "This module contains a collection of YANG definitions for  
        configuring network interfaces.  
  
        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.: update the date below with the date of RFC publication
// and remove this note.
revision 2011-05-20 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Interface Configuration";
}

/* Typedefs */

typedef interface-ref {
  type leafref {
    path "/if:interfaces/if:interface/if:name";
  }
  description
    "This type is used by data models that need to reference
    interfaces.";
}

/* Features */

feature snmp-if-mib {
  description
    "This feature indicates that the server implements IF-MIB.";
  reference
    "RFC 2863: The Interfaces Group MIB";
}

/* Data nodes */

container interfaces {
  description
    "Interface parameters.";

  list interface {
    key "name";
    unique "type location";

    description
      "The list of configured interfaces on the device.";

    leaf name {
      type string {
        length "1..255";
      }
    }
  }
}
```

```
}
description
  "An arbitrary name for the interface.

  A device MAY restrict the allowed values for this leaf,
  possibly depending on the type and location.

  For example, if a device has a single array of 8 ethernet
  ports, the name might be restricted to be on the form
  'ethN', where N is an integer between '1' and '8'."
}

leaf description {
  type string;
  description
    "A textual description of the interface.

    This leaf MAY be mapped to ifAlias by an implementation.
    Such an implementation MAY restrict the length of the
    value of this leaf so that it matches the restrictions
    of ifAlias."
  reference
    "RFC 2863: The Interfaces Group MIB - ifAlias";
}

leaf type {
  type ift:iana-if-type;
  mandatory true;
  description
    "The type of the interface.

    When an interface entry is created, a server MAY
    initialize the type leaf with a valid value, e.g., if it
    is possible to derive the type from the name of the
    interface."
}

leaf location {
  type string;
  description
    "The device-specific location of the interface of a
    particular type. The format of the location string
    depends on the interface type and the device.

    Media-specific modules must specify if the location
    is needed for the given type.

    For example, if a device has a single array of 8 ethernet
```

ports, the location can be one of '1' to '8'. As another example, if a device has N cards of M ports, the location can be on the form 'n/m'.

When an interface entry is created, a server MAY initialize the location leaf with a valid value, e.g., if it is possible to derive the location from the name of the interface."

}

leaf enabled {

  type boolean;

  default "true";

  description

    "The desired state of the interface.

    This leaf contains the configured, desired state of the interface. Systems that implement the IF-MIB use the value of this leaf to set IF-MIB.ifAdminStatus after an ifEntry has been initialized, as described in RFC 2863."

  reference

    "RFC 2863: The Interfaces Group MIB - ifAdminStatus";

}

leaf-list if-index {

  if-feature snmp-if-mib;

  type int32 {

    range "1..2147483647";

  }

  config false;

  description

    "The list of ifIndex values for all ifEntries that are represented by this interface. If there is a one-to-one mapping between the interface and entries in the ifTable, this leaf-list will have a single value.

    Media-specific modules must specify how the type is mapped to entries in the ifTable."

  reference

    "RFC 2863: The Interfaces Group MIB - ifIndex";

}

leaf mtu {

  type uint32;

  description

    "The size, in octets, of the largest packet that the interface can send and receive. This node might not be valid for all interface types.

```
        Media-specific modules must specify any restrictions on
        the mtu for their interface type.";
    }

    leaf link-up-down-trap-enable {
        if-feature snmp-if-mib;
        type enumeration {
            enum enabled {
                value 1;
            }
            enum disabled {
                value 2;
            }
        }
        description
            "Indicates whether linkUp/linkDown SNMP traps should be
            generated for this interface.

            If this node is not configured, the value 'enabled' is
            operationally used by the server for interfaces which do
            not operate on top of any other interface (as defined in
            the ifStackTable), and 'disabled' otherwise.";
        reference
            "RFC 2863: The Interfaces Group MIB -
            ifLinkUpDownTrapEnable";
    }
}
}
}

<CODE ENDS>
```

## 5. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made.

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

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-interfaces
namespace:	urn:ietf:params:xml:ns:yang:ietf-interfaces
prefix:	if
reference:	RFC XXXX

## 6. Security Considerations

The YANG module and submodules defined in this memo are designed to be accessed via the NETCONF protocol [I-D.ietf-netconf-4741bis]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [I-D.ietf-netconf-rfc4742bis].

There are a number of data nodes defined in the YANG module and submodules which are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/interfaces/interface: This list specify the configured interfaces on a device. Unauthorized access to this list could cause the device to ignore packets destined to it.

/interfaces/interface/enabled: This leaf controls if an interface is enabled or not. Unauthorized access to this leaf could cause the device to ignore packets destined to it.

## 7. Acknowledgments

The author wishes to thank Per Hedeland, Ladislav Lhotka, and Juergen Schoenwaelder for their helpful comments.



## 8. Normative References

- [I-D.ietf-netconf-4741bis]  
Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", draft-ietf-netconf-4741bis-10 (work in progress), March 2011.
- [I-D.ietf-netconf-rfc4742bis]  
Wasserman, M. and T. Goddard, "Using the NETCONF Configuration Protocol over Secure Shell (SSH)", draft-ietf-netconf-rfc4742bis-08 (work in progress), March 2011.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

#### Appendix A. Example: Ethernet Interface Module

This section gives a simple example of how an Ethernet interface module could be defined. It demonstrates how media-specific configuration parameters can be conditionally augmented to the generic interface list. It is not intended as a complete module for ethernet configuration.

```
module ex-ethernet {
  namespace "http://example.com/ethernet";
  prefix "eth";

  import ietf-interfaces {
    prefix if;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'ethernetCsmacd'";

    container ethernet {
      must "../if:location" {
        description
          "An ethernet interface must specify the physical location
           of the ethernet hardware.";
      }
      choice transmission-params {
        case auto {
          leaf auto-negotiate {
            type empty;
          }
        }
        case manual {
          leaf duplex {
            type enumeration {
              enum "half";
              enum "full";
            }
          }
          leaf speed {
            type enumeration {
              enum "10Mb";
              enum "100Mb";
              enum "1Gb";
              enum "10Gb";
            }
          }
        }
      }
    }
  }
  // other ethernet specific params...
}
```

## Appendix B. Example: Ethernet Bonding Interface Module

This section gives an example of how interface layering can be defined. An ethernet bonding interface is defined, which bonds several ethernet interfaces into one logical interface.

```
module ex-ethernet-bonding {
  namespace "http://example.com/ethernet-bonding";
  prefix "bond";

  import ietf-interfaces {
    prefix if;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'ieee8023adLag'";

    leaf-list slave-if {
      type if:interface-ref;
      must "/if:interfaces/if:interface[if:name = current()]"
        + "/if:type = 'ethernetCsmacd'" {
        description
          "The type of a slave interface must be ethernet.";
      }
    }
    leaf bonding-mode {
      type enumeration {
        enum round-robin;
        enum active-backup;
        enum broadcast;
      }
    }
    // other bonding config params, failover times etc.
  }
}
```

## Appendix C. Example: VLAN Interface Module

This section gives an example of how a vlan interface module can be defined.

```
module ex-vlan {
  namespace "http://example.com/vlan";
  prefix "vlan";

  import ietf-interfaces {
    prefix if;
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'ethernetCsmacd' or
         if:type = 'ieee8023adLag'";
    leaf vlan-tagging {
      type boolean;
      default false;
    }
  }

  augment "/if:interfaces/if:interface" {
    when "if:type = 'l2vlan'";

    leaf base-interface {
      type if:interface-ref;
      must "/if:interfaces/if:interface[if:name = current()]"
        + "/vlan:vlan-tagging = true" {
        description
          "The base interface must have vlan tagging enabled.";
      }
    }
    leaf vlan-id {
      type uint16 {
        range "1..4094";
      }
      must "../base-interface";
    }
  }
}
```

## Appendix D. Example: IP Module

This section gives an example how an IP module can be defined.

```
module ex-ip {  
  
  namespace "http://example.com/ip";  
  prefix "ip";  
  
  import ietf-interfaces {  
    prefix if;  
  }  
  
  import ietf-inet-types {  
    prefix inet;  
  }  
  
  augment "/if:interfaces/if:interface" {  
    container ip {  
      list address {  
        key "ip";  
        leaf ip {  
          type inet:ip-address;  
        }  
        leaf prefix-length {  
          type uint16;  
          // range depends on type of address  
        }  
      }  
    }  
  }  
}
```

## Appendix E. Example: NETCONF &lt;get&gt; reply

This section gives an example of a reply to the NETCONF <get> request for a device that implements the example data models above.

```
<rpc-reply
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="101">
  <data>
    <interfaces
      xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type>ethernetCsmacd</type>
        <location>0</location>
        <admin-status>up</admin-status>
        <if-index>2</if-index>
        <ip xmlns="http://example.com/ip">
          <address>
            <ip>192.0.2.1</ip>
            <prefix-length>24</prefix-length>
          </address>
        </ip>
      </interface>
      <interface>
        <name>eth1</name>
        <type>ethernetCsmacd</type>
        <location>1</location>
        <admin-status>up</admin-status>
        <if-index>7</if-index>
        <ip xmlns="http://example.com/ip">
          <address>
            <ip>192.168.1.1</ip>
            <prefix-length>24</prefix-length>
          </address>
        </ip>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

## Appendix F. ChangeLog

RFC Editor: remove this section upon publication as an RFC.

## F.1. Version -01

- o Changed leaf "if-admin-status" to leaf "enabled".
- o Added Security Considerations



Author's Address

Martin Bjorklund  
Tail-f Systems

Email: [mbj@tail-f.com](mailto:mbj@tail-f.com)



NETMOD  
Internet-Draft  
Intended status: Standards Track  
Expires: October 29, 2011

L. Lhotka  
CESNET  
April 27, 2011

A YANG Data Model for Routing Configuration  
draft-ietf-netmod-routing-cfg-00

Abstract

This document contains a specification of two YANG modules that together provide a data model for essential configuration of a routing subsystem. It is expected that this module will serve as a basis for further development of data models for individual routing protocols and other related functions. The present data model defines the building blocks for such configurations - routing processes, routes and routing tables, routing protocol instances and route filters.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 29, 2011.

Copyright Notice

Copyright (c) 2011 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.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology and Notation . . . . .	4
2.1. Glossary of New Terms . . . . .	4
2.2. Prefixes in Data Node Names . . . . .	5
3. Objectives . . . . .	6
4. The Design of the Core Routing Data Model . . . . .	7
4.1. Route . . . . .	9
4.2. Routing Tables . . . . .	9
4.3. Routing Protocol Instances . . . . .	10
4.3.1. Defining New Routing Protocols . . . . .	11
4.4. Route Filters . . . . .	13
4.5. RPC Operations . . . . .	13
5. Routing YANG Module . . . . .	15
6. IPv4 Unicast Routing YANG Module . . . . .	24
7. IANA Considerations . . . . .	33
8. Security Considerations . . . . .	34
9. Acknowledgments . . . . .	35
10. References . . . . .	36
10.1. Normative References . . . . .	36
10.2. Informative References . . . . .	36
Appendix A. Example - Adding a New Routing Protocol . . . . .	37
A.1. Example YANG Module for Routing Information Protocol . . . . .	37
A.2. Sample Reply to the NETCONF <get> Message . . . . .	38
Author's Address . . . . .	44

## 1. Introduction

This document contains an initial specification of two YANG modules, "ietf-routing" and "ietf-ipv4-unicast-routing", that together define the so-called core routing data model. This data model will serve as a basis for the development of data models for more sophisticated routing configurations. While these two modules can be directly used for simple IPv4-only devices with static routing, their main purpose is to provide basic building blocks for more complicated setups involving other address families such as IPv6, multiple routing protocols, and advanced functions, for example route filtering and policy routing. To this end, it is expected that this module will be augmented by numerous modules developed by other IETF working groups.

## 2. Terminology and Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The following terms are defined in [RFC4741]:

- o client
- o message
- o operation
- o server

The following terms are defined in [RFC6020]:

- o augment
- o configuration data
- o container
- o data model
- o data node
- o data type
- o identity
- o mandatory node
- o module
- o operational state data
- o prefix
- o RPC operation

### 2.1. Glossary of New Terms

- o active route: a route which is actually used for packet forwarding. If there are multiple candidate routes with the same destination prefix, then it is up to the routing algorithm to select the active route.

## 2.2. Prefixes in Data Node Names

In this document, names of data nodes are used mostly without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed with the standard prefixes associated with YANG modules, as shown in Table 1.

Prefix	YANG module	Reference
eth	ex-ethernet	[YANG-IF]
if	ietf-interfaces	[YANG-IF]
inet	ietf-inet-types	[RFC6021]
ip	ex-ip	[YANG-IF]
rip	example-rip	Appendix A
rt	ietf-routing	Section 5
v4ur	ietf-ipv4-unicast-routing	Section 6
yang	ietf-yang-types	[RFC6021]

Table 1: Prefixes and corresponding YANG modules

### 3. Objectives

The initial design of the core routing data model was driven by the following main objectives:

- o The data model should be suitable for the common address families, in particular IPv4 and IPv6, and for unicast and multicast routing as well as Multiprotocol Label Switching (MPLS).
- o Simple routing setups, such as static routing, should be configurable in a simple way, ideally without any need to develop additional YANG modules.
- o On the other hand, the core routing framework must allow for complicated setups involving multiple routing tables and multiple routing protocols, as well as controlled redistributions of routing information.
- o Device vendors will want to map the data models built on this generic framework to their proprietary data models and configuration interfaces. Therefore, the framework should be flexible enough to facilitate such a mapping and accommodate data models with different logic.



#### 4. The Design of the Core Routing Data Model

The core routing data model consists of two YANG modules. The first module, "ietf-routing", is rather minimal and provides only a top-level container ("routing") and a list of routing processes. Each routing process represents an instance of a (virtual) router with a separate forwarding table (FIB, forwarding information base). For a given address family, specified by an Address Family Identifier (AFI) [IANA-AFI] and Subsequent Address Family Identifier (SAFI) [IANA-SAFI], several independent routing processes may be configured.

The second YANG module, "ietf-ipv4-unicast-routing", provides a data modeling framework for IPv4 unicast routing with several essential components: routes, routing tables, routing protocol instances, route filters and RPC operations. The following subsections provide further details about these components.

By combining the components in various ways, and possibly filling them with appropriate contents defined in other modules, a broad range of routing setups can be covered.

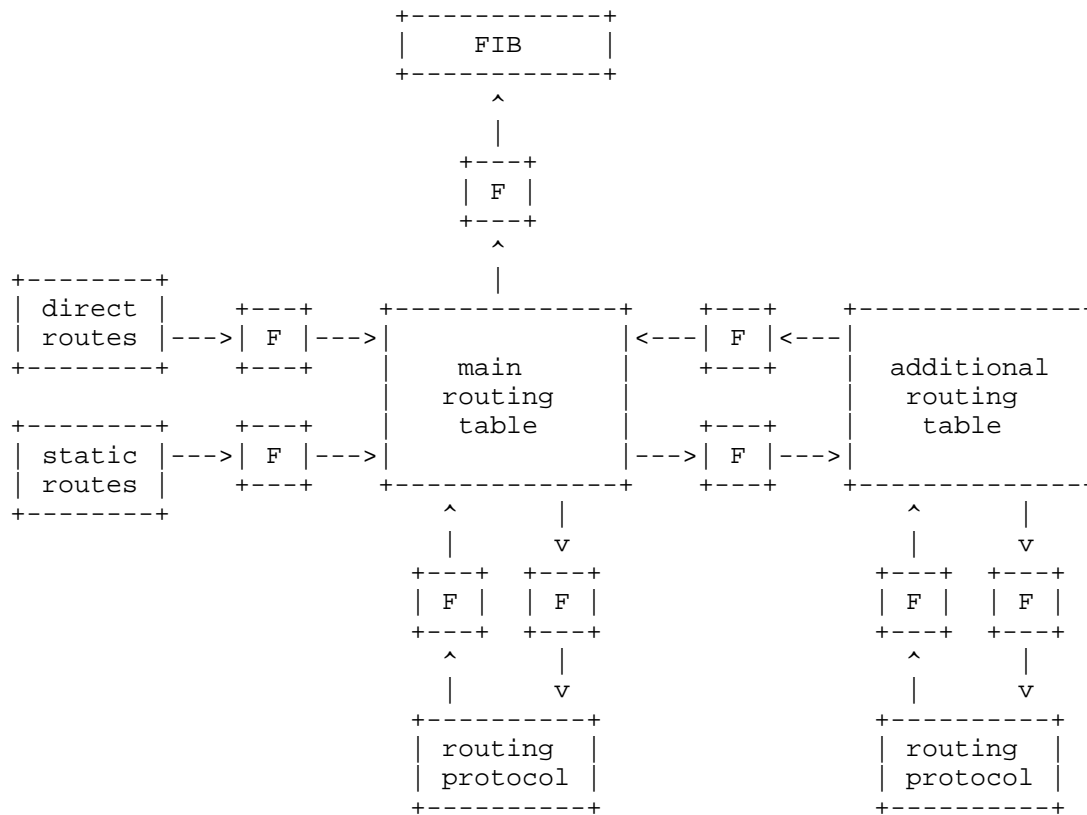


Figure 1: Example setup of the routing subsystem

Figure 1 shows an example of a more complicated setup:

- o Along with the main routing table, which must always be present, an additional routing table is defined.
- o Each routing protocol instance, including the "static" and "direct" pseudo-protocol instances, is connected to exactly one routing table with which it can exchange routes (in both directions, except for the "static" and "direct" pseudo-protocols).
- o Routing tables may also be connected to each other and exchange routes in one or both directions.
- o The forwarding information base (FIB) is a special routing table which must always be present. Typically, the FIB receives the active routes from the main routing table and the operating system

kernel uses this information for packet forwarding.

- o Route exchanges along all connections may be controlled by means of route filters, denoted by "F" in the figure.

#### 4.1. Route

Routes are basic units of information in a routing system. The "ietf-ipv4-unicast-routing" module defines only the following minimal set of route attributes:

- o destination-prefix - IP prefix specifying the set of destination addresses for which the route may be used. This attribute is mandatory.
- o next-hop - IP address of the adjacent router or host to which packets with destination addresses belonging to destination-prefix should be sent.
- o outgoing-interface - network interface that should be used for sending packets with destination addresses belonging to destination-prefix.

The above list of route attributes is sufficient for a simple static routing configuration. It is expected that future modules defining routing protocols will add other route attributes such as metrics or preferences.

Routes and their attributes are used in both configuration data, for example as manually configured static routes, as well as in operational state data, for example as entries in routing tables.

#### 4.2. Routing Tables

Routing tables are lists of routes complemented with administrative data, namely:

- o source-protocol - name of the routing protocol from which the route was originally obtained.
- o last-modified - date and time of last modification, or installation, of the route.

In the core routing data model, the list of routes in routing tables is represented as operational state data. Routing protocol operations result in route additions, removals and modifications. This also includes manipulations via the "static" pseudo-protocol.

The "ietf-ipv4-unicast-routing" module requires that at least the following two routing tables MUST be configured for each routing process:

- o The "ipv4-unicast-fib" table is the forwarding information base used by the operating system kernel for forwarding IPv4 unicast datagrams.
- o The "ipv4-unicast-main" table is the main routing table. By default, all IPv4 unicast routing protocols exchange routes with this table, and active routes from the "ipv4-unicast-main" routing table are installed in the "ipv4-unicast-fib" table and used for packet forwarding.

Additional routing tables MAY be configured.

Every routing table MAY serve as a source of routes for other routing tables. To achieve this, one or more recipient routing tables MAY be specified in the configuration of the source routing table. In addition, a route filter may be configured for each recipient routing table, which selects and/or manipulates the routes that are passed on between the source and recipient routing table.

#### 4.3. Routing Protocol Instances

The "ietf-ipv4-unicast-routing" module provides an open-ended framework for defining multiple routing protocol instances. Each of them is identified by a name, which is unique within a routing process, and MUST be assigned a type from a selection which includes all routing protocol types supported by the server, such as RIP, OSPF or BGP.

Each routing protocol instance is connected to exactly one routing table. By default, every routing protocol instance is connected to the main routing table, but any routing protocol instance can be configured to use a different routing table, provided such an extra table is configured.

Routes learned from the network by a routing protocol instance are passed to the connected routing table and vice versa - routes appearing in a routing table are passed to all routing protocol connected to the table and advertised by that protocol to the network.

Two independent route filters (see Section 4.4) may be defined for a routing protocol instance to control the exchange of routes in both directions between the routing protocol instance and the connected routing table:

- o import filter controls which routes are passed from a routing protocol instance to the routing table,
- o export filter controls which routes the routing protocol instance may receive from the connected routing table.

Note that, for historical reasons, the terms import and export are used from the viewpoint of a routing table.

The "ietf-ipv4-unicast-routing" module defines two special routing protocols - "direct" and "static". Both are in fact pseudo-protocols, which means that they are confined to the local device and do not exchange any routing information with neighboring routers. Routes from both "direct" and "static" protocol instances are passed to the connected routing table (subject to route filters, if any), but an exchange in the opposite direction is not allowed.

Every routing process MUST contain exactly one instance of the "direct" pseudo-protocol. It is the source of routes to directly connected networks (so-called direct routes). Such routes are supplied by the operating system kernel based on the detected and configured network interfaces, and they usually appear in the main routing table. However, using the framework defined in this document, the target routing table for direct routes can be changed by connecting the "direct" protocol instance to a non-default routing table, and the direct routes can also be filtered before they appear in the routing table.

The "static" routing pseudo-protocol allows for specifying routes manually. It can be configured in zero or more instances, although typically one instance suffices.

#### 4.3.1. Defining New Routing Protocols

It is expected that future YANG modules will create data models for additional routing protocol types. In order to do so, the new module has to define the protocol-specific information and fit it to the core routing framework in the following way:

- o A new identity MUST be defined for the routing protocol and its base identity set to "rt:routing-protocol", or to an identity derived from "rt:routing-protocol".
- o Additional route attributes MAY be defined. Their definitions have to be inserted as operational state data by augmenting the definition of "v4ur:route" inside "v4ur:routing-table". Naturally, route attributes (including the extra attributes) may be used in configuration data, too, as demonstrated by the

```
"static" pseudo-protocol.
```

- o The recommended way of defining configuration data specific to the new protocol is to augment the "routing-protocol-instance" list entry with a container that encapsulates the configuration hierarchy of the new protocol. The "augment" statement SHOULD be made conditional by using a "when" substatement requiring that the new nodes be used only if the "type" leaf node is equal to the new protocol's identity.

The above steps are implemented by the example YANG module for the RIP routing protocol in Appendix A. First, the module defines a new identity for the RIP protocol:

```
identity rip {  
  base rt:routing-protocol;  
  description "Identity for the RIP routing protocol.";  
}
```

Second, new route attributes specific for the RIP protocol ("metric" and "tag") are added:

```
augment "/rt:routing/rt:routing-process/v4ur:ipv4-unicast-routing/"  
  + "v4ur:routing-tables/v4ur:routing-table/"  
  + "v4ur:routes/v4ur:route" {  
    when "../../../v4ur:routing-protocol-instances/"  
      + "v4ur:routing-protocol-instance[rt:name=" "  
        + "current()/v4ur:source-protocol]/v4ur:type='rip:rip'";  
    description  
      "RIP-specific route components.";  
    leaf metric { ... }  
    leaf tag { ... }  
  }
```

The "when" statement is used to make sure that the new route attributes are only valid when the source protocol is RIP.

Finally, RIP-specific configuration data are integrated into the "v4ur:routing-protocol-instance" node by using the following "augment" statement, which applies only to routing protocol instances whose type is "rip:rip", and which is a part of a routing process whose address family is "IPv4" and subsequent address family identifier is "nlri-unicast":

```
augment "/rt:routing/rt:routing-process/v4ur:ipv4-unicast-routing/"
  + "v4ur:routing-protocol-instances/"
  + "v4ur:routing-protocol-instance" {
    when "v4ur:type = 'rip:rip' and ../../../../rt:address-family = 'ipV4'"
      + " and ../../../../safi = 'nlri-unicast'";
    container rip-configuration {
      ...
    }
  }
}
```

#### 4.4. Route Filters

The "ietf-ipv4-unicast-routing" module provides a skeleton for defining route filters that can be used to restrict the set of routes being exchanged between a routing protocol instance and a routing table, or between a source and a recipient routing table. Route filters may also manipulate routes, i.e., add, delete, or modify their properties.

By itself, the route filtering framework defined in the "ietf-ipv4-unicast-routing" module allows to establish only the two extreme routing policies in which either all routes are allowed or all routes are denied. It is expected that a real route filtering framework (or several alternative frameworks) will be developed separately.

Each route filter is identified by a name which is unique within a routing process. Its type MUST be specified by the "type" identity reference - this opens the space for multiple route filtering framework implementations. The default value for route filter type is the identity "deny-all-route-filter" defined in the "ietf-routing" module, which represents the "deny all" route filtering policy.

#### 4.5. RPC Operations

The "ietf-ipv4-unicast-routing-module" defines two RPC operations:

- o "delete-route" operations allows the client to immediately delete specific route(s) from a routing table within a routing process. The first input parameter of this operation is the name of the routing process, the second parameter is the routing table to act upon, and the third (optional) parameter is the "route" container with zero or more of the following route attributes: "destination-prefix", "next-hop" and "outgoing-interface". All routes that match these attributes MUST be deleted from the selected routing table. If the "route" container is missing or empty, all routes from the selected routing table MUST be deleted.

- o "get-route" is used for querying the forwarding information base of a routing process. The first input parameter is the name of a routing process whose FIB is to be queried, and the second parameter is an IPv4 destination address. The server replies with an active route which is used for forwarding datagrams to the destination address within the selected routing process.



## 5. Routing YANG Module

```
<CODE BEGINS> file "ietf-routing@2011-04-27.yang"
```

```
module ietf-routing {
  namespace "urn:ietf:params:xml:ns:yang:ietf-routing";
  prefix rt;

  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:   Ladislav Lhotka
    <mailto:lhodka@cesnet.cz>";
  description
    "This module contains YANG definitions for top-level containers
    for the configuration of routing together with several type
    definitions and identities.";

  revision 2011-04-27 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for Routing Configuration";
  }

  /* Identities */

  identity routing-protocol {
    description
      "Base identity from which routing protocol identities are
      derived.";
  }

  identity direct {
    base routing-protocol;
    description
      "Identity for the pseudo-protocol providing routes to directly
      connected networks. An implementation MUST preconfigure
      exactly one instance of this pseudo-protocol for each routing
      process."; }
}
```

```
identity static {
  base routing-protocol;
  description
    "Identity for static routing pseudo-protocol.";
}

identity route-filter {
  description
    "Base identity from which all route filters are
    derived.";
}

identity deny-all-route-filter {
  base route-filter;
  description
    "This identity represents a route filter that blocks all
    routes.";
}

/* Type definitions */

typedef address-family {
  type enumeration {
    enum "other" {
      value 0;
      description
        "none of the following";
    }
    enum "ipV4" {
      value 1;
      description
        "IP Version 4";
    }
    enum "ipV6" {
      value 2;
      description
        "IP Version 6";
    }
    enum "nsap" {
      value 3;
      description
        "NSAP";
    }
    enum "hdlc" {
      value 4;
      description
        "(8-bit multidrop)";
    }
  }
}
```

```
enum "bbn1822" {
    value 5;
    description
        "BBN Report 1822";
}
enum "all802" {
    value 6;
    description
        "(includes all 802 media plus Ethernet 'canonical
        format')";
}
enum "e163" {
    value 7;
}
enum "e164" {
    value 8;
    description
        "(SMDS, FrameRelay, ATM)";
}
enum "f69" {
    value 9;
    description
        "(Telex)";
}
enum "x121" {
    value 10;
    description
        "(X.25, Frame Relay)";
}
enum "ipx" {
    value 11;
    description
        "IPX (Internet Protocol Exchange)";
}
enum "appleTalk" {
    value 12;
    description
        "Apple Talk";
}
enum "decnetIV" {
    value 13;
    description
        "DEC Net Phase IV";
}
enum "banyanVines" {
    value 14;
    description
        "Banyan Vines";
}
```

```
}
enum "e164withNsap" {
  value 15;
  description
    "(E.164 with NSAP format subaddress)";
}
enum "dns" {
  value 16;
  description
    "(Domain Name System)";
}
enum "distinguishedName" {
  value 17;
  description
    "(Distinguished Name, per X.500)";
}
enum "asNumber" {
  value 18;
  description
    "(16-bit quantity, per the AS number space)";
}
enum "xtpOverIPv4" {
  value 19;
  description
    "XTP over IP version 4";
}
enum "xtpOverIpv6" {
  value 20;
  description
    "XTP over IP version 6";
}
enum "xtpNativeModeXTP" {
  value 21;
  description
    "XTP native mode XTP";
}
enum "fibreChannelWWPN" {
  value 22;
  description
    "Fibre Channel World-Wide Port Name";
}
enum "fibreChannelWWNN" {
  value 23;
  description
    "Fibre Channel World-Wide Node Name";
}
enum "gwid" {
  value 24;
```

```
        description
            "Gateway Identifier";
    }
    enum "afi" {
        value 25;
        description
            "AFI for L2VPN";
    }
}
description
    "This typedef is a YANG enumeration of IANA-registered
    address families.";
reference
    "http://www.iana.org/assignments/ianaaddressfamilynumbers-mib";
}

typedef subsequent-address-family {
    type enumeration {
        enum "nlri-unicast" {
            value 1;
            description
                "Network Layer Reachability Information used for
                unicast forwarding";
            reference "RFC4760";
        }
        enum "nlri-multicast" {
            value 2;
            description
                "Network Layer Reachability Information used for
                multicast forwarding";
            reference "RFC4760";
        }
        enum "nlri-mpls" {
            value 4;
            description
                "Network Layer Reachability Information (NLRI) with
                MPLS Labels";
            reference "RFC3107";
        }
        enum "mcast-vpn" {
            value 5;
            description
                "MCAST-VPN";
            reference "draft-ietf-l3vpn-2547bis-mcast-bgp-08";
        }
        enum "nlri-dynamic-ms-pw" {
            value 6;
            status obsolete;
        }
    }
}
```

```
description
  "Network Layer Reachability Information used for Dynamic
  Placement of Multi-Segment Pseudowires (TEMPORARY -
  Expires 2008-08-23)";
reference "draft-ietf-pwe3-dynamic-ms-pw-13";
}
enum "tunnel-safi" {
  value 64;
  description
    "Tunnel SAFI";
  reference "draft-nalawade-kapoor-tunnel-safi-05";
}
enum "vpls" {
  value 65;
  description
    "Virtual Private LAN Service (VPLS)";
  reference "RFC4761, RFC6074";
}
enum "bgp-mdt" {
  value 66;
  description
    "BGP MDT SAFI";
  reference "RFC6037";
}
enum "bgp-4over6" {
  value 67;
  description
    "BGP 4over6 SAFI";
  reference "RFC5747";
}
enum "bgp-6over4" {
  value 68;
  description
    "BGP 6over4 SAFI";
  reference "mailto:cuiyong&tsinghua.edu.cn";
}
enum "llvpn-auto-discovery" {
  value 69;
  description
    "Layer-1 VPN auto-discovery information";
  reference "draft-ietf-llvpn-bgp-auto-discovery-05";
}
enum "mpls-vpn" {
  value 128;
  description
    "MPLS-labeled VPN address";
  reference "RFC4364";
}
```

```
enum "multicast-bgp-mpls-vpn" {
  value 129;
  description
    "Multicast for BGP/MPLS IP Virtual Private Networks
    (VPNs)";
  reference
    "draft-ietf-l3vpn-2547bis-mcast-10,
    draft-ietf-l3vpn-2547bis-mcast-10";
}
enum "route-target-constraints" {
  value 132;
  description
    "Route Target constraints";
  reference "RFC4684";
}
enum "ipv4-diss-flow" {
  value 133;
  description
    "IPv4 dissemination of flow specification rules";
  reference "RFC5575";
}
enum "vpn4-diss-flow" {
  value 134;
  description
    "IPv4 dissemination of flow specification rules";
  reference "RFC5575";
}
enum "vpn-auto-discovery" {
  value 140;
  description
    "VPN auto-discovery";
  reference "draft-ietf-l3vpn-bgpvpn-auto-09";
}
}
description
  "This typedef is a YANG enumeration of IANA-registered
  subsequent address families.";
reference "http://www.iana.org/assignments/safi-namespace/"
  + "safi-namespace.xml";
}

typedef routing-process-ref {
  type leafref {
    path "/rt:routing/rt:routing-process/rt:name";
  }
}
description
  "This type is used for leafs that reference a routing
  process.";
```

```
}

/* Data nodes */

container routing {
  description
    "Routing parameters.";
  list routing-process {
    key "name";
    description
      "Each entry is a container for configuration and operational
      state data of a single (virtual) router for a given address
      family and subsequent address family identifier (SAFI). Each
      entry has a unique name.

      The definitions of data for a particular address family and
      subsequent address family shall be provided via augmentation
      by other modules.";
    leaf name {
      type string;
      description
        "The unique name of the routing process.";
    }
    leaf address-family {
      type address-family;
      default "ipV4";
      description
        "Address family of the routing process.";
    }
    leaf safi {
      type subsequent-address-family;
      default "nlri-unicast";
      description
        "Subsequent address family identifier of the routing
        process.";
    }
    leaf description {
      type string;
      description
        "Textual description of the routing process.";
    }
    leaf enabled {
      type boolean;
      default "true";
      description
        "Enable or disable the routing process. The default value
        is 'true', which means that the process is enabled.";
    }
  }
}
```



```
    }  
  }  
}
```

<CODE ENDS>

## 6. IPv4 Unicast Routing YANG Module

```
<CODE BEGINS> file "ietf-ipv4-unicast-routing@2011-04-27.yang"
```

```
module ietf-ipv4-unicast-routing {  
  namespace "urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing";  
  prefix v4ur;  
  
  import ietf-routing {  
    prefix rt;  
  }  
  import ietf-yang-types {  
    prefix yang;  
  }  
  import ietf-inet-types {  
    prefix inet;  
  }  
  import ietf-interfaces {  
    prefix if;  
  }  
}
```

```
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: Ladislav Lhotka
```

```
<mailto:lhotka@cesnet.cz>";
```

```
description
```

```
"This module augments the 'ietf-routing' module with YANG  
definitions for basic configuration of IPv4 unicast routing.
```

```
It is immediately usable for a device that needs just a single  
routing table populated with static routes.
```

```
On the other hand, the framework is designed to handle  
arbitrarily complex configurations with any number of routing  
tables and various routing protocols (in multiple instances).";
```

```
revision 2011-04-27 {
```

```
  description
```

```
    "Initial revision.";
```

```
    reference
      "RFC XXXX: A YANG Data Model for Routing Configuration";
  }

/* Groupings */

grouping routing-process-name {
  leaf routing-process-name {
    type rt:routing-process-ref;
    must "/rt:routing/rt:routing-process[rt:name = current()]"
      + "/rt:address-family = 'IPv4' and "
      + "/rt:routing/rt:routing-process[rt:name = current()]"
      + "/rt:safi = 'nlri-unicast'" {
      description
        "The referred routing process must be IPv4 unicast.";
    }
  }
  description "The name of a routing process.";
}
description
  "This grouping defines the first common parameter of both
  RPC operations below.";
}

/* RPC operations */

rpc get-route {
  description
    "Query the forwarding information base of an IPv4 unicast
    routing process whose name is given as the first
    parameter. The second parameter is an IPv4 destination
    address. The server returns the route which is currently used
    for forwarding datagrams to that destination address, or an
    error message, if no such route exists.";
  input {
    uses routing-process-name;
    leaf destination-address {
      type inet:ipv4-address;
      description
        "Second parameter - IPv4 destination address.";
    }
  }
  output {
    container route {
      description
        "Contents of the reply.";
      leaf destination-prefix {
        type inet:ipv4-prefix;
        mandatory true;
      }
    }
  }
}
```

```
        description
          "Destination prefix of the returned route.";
      }
      leaf next-hop {
        type inet:ipv4-address;
        description
          "Next hop address of the returned route.";
      }
      leaf outgoing-interface {
        type if:interface-ref;
        description
          "Outgoing interface of the returned route.";
      }
    }
  }
}

rpc delete-route {
  description

    "Delete all routes that match the given attributes from a
    routing table within a routing process.

    Parameters:
    1. routing process name,
    2. routing table name,
    3. Container 'route' with route attributes.

    <ok> is returned by the server upon successful completion.";

  input {
    uses routing-process-name;
    leaf routing-table {
      type leafref {
        path "/rt:routing/rt:routing-process[rt:name=current()]/../"
          + "routing-process-name[/ipv4-unicast-routing/"
          + "routing-tables/routing-table/name";
      }
      mandatory true;
      description
        "First parameter.";
    }
    container route {
      description
        "Second parameter. All routes matching the route
        attributes must be deleted from the routing table.

        If this container is empty or missing, all routes
```

```

        from the selected routing table are deleted.";
    leaf destination-prefix {
        type inet:ipv4-prefix;
        description
            "Match destination prefix.";
    }
    leaf next-hop {
        type inet:ipv4-address;
        description
            "Match next hop.";
    }
    leaf outgoing-interface {
        type if:interface-ref;
        description
            "Match outgoing interface.";
    }
}
}
}

/* Data nodes */

augment "/rt:routing/rt:routing-process" {
    when "afi='IPv4' and safi='nlri-unicast'" {
        description
            "IPv4 unicast.";
    }
    description
        "Definitions of data nodes that augment a routing process
        for IPv4 unicast.";
    container ipv4-unicast-routing {
        description
            "Container for IPv4 unicast routing configuration and
            operational state data.";
        container routing-protocol-instances {
            description
                "Container for the list of configured routing protocol
                instances.";
            list routing-protocol-instance {
                key "name";
                description
                    "An instance of a routing protocol.";
                container static-routes {
                    when "../type='rt:static'" {
                        description
                            "These data nodes are only valid for the static
                            pseudo-protocol.";
                    }
                }
            }
        }
    }
}

```

```
    }
    description
      "Configuration of a 'static' pseudo-protocol
       instance consists of a list of routes.";
    list static-route {
      key "id";
      ordered-by user;
      description
        "An user-ordered list of static routes.";
      leaf id {
        type string;
        description
          "An identification string for the route.";
      }
      leaf description {
        type string;
        description
          "Textual description of the route.";
      }
      leaf destination-prefix {
        type inet:ipv4-prefix;
        mandatory true;
        description
          "The destination prefix for which the route may
           be used.";
      }
      leaf next-hop {
        type inet:ipv4-address;
        description
          "IPv4 address of the host or router to which
           packets whose address matches 'destination-prefix'
           are to be forwarded.";
      }
      leaf outgoing-interface {
        type if:interface-ref;
        description
          "Name of the outgoing interface. This attribute
           is mainly used in direct routes.";
      }
    }
  }
  leaf name {
    type string;
    description
      "The name of the routing protocol instance.";
  }
  leaf description {
    type string;
```

```
    description
      "Textual description of the routing protocol
      instance.";
  }
  leaf type {
    type identityref {
      base rt:routing-protocol;
    }
    mandatory true;
    description
      "Type of the routing protocol - an identity derived
      from the 'rt:routing-protocol' base identity.";
  }
  leaf routing-table {
    type leafref {
      path "../../../../../routing-tables/routing-table/name";
    }
    default "ipv4-unicast-main";
    description
      "The routing table to which the routing protocol
      instance is connected. By default it is the
      'ipv4-unicast-main' table.";
  }
  leaf import-filter {
    type leafref {
      path "../../../../../route-filters/route-filter/name";
    }
    description
      "Reference to a route filter that is used for
      filtering routes passed from this routing protocol
      instance to the routing table specified by the
      'routing-table' sibling node. If this leaf is not
      present, the behavior is protocol-specific, but
      typically it means that all routes are accepted.";
  }
  leaf export-filter {
    type leafref {
      path "../../../../../route-filters/route-filter/name";
    }
    description
      "Reference to a route filter that is used for filtering
      routes passed from the routing table specified by the
      'routing-table' sibling to this routing protocol
      instance. If this leaf is not present, the behavior is
      protocol-specific - typically it means that all routes
      are accepted, except for the 'direct' and 'static'
      pseudo-protocols which accept no routes from any
      routing table.";
```

```
    }  
  }  
}  
container route-filters {  
  description  
    "Container for configured route filters.";  
  list route-filter {  
    key "name";  
    description  
      "Route filters are used for filtering and/or manipulating  
      routes that are passed between a routing protocol and a  
      routing table or vice versa, or between two routing  
      tables. It is expected that other modules augment this  
      list with contents specific for a particular route  
      filter type.";  
    leaf name {  
      type string;  
      description  
        "The name of the route filter.";  
    }  
    leaf description {  
      type string;  
      description  
        "Textual description of the route filter.";  
    }  
    leaf type {  
      type identityref {  
        base rt:route-filter;  
      }  
      default "rt:deny-all-route-filter";  
      description  
        "Type of the route-filter - an identity derived  
        from the 'rt:route-filter' base identity. The default  
        value represents an all-blocking filter.";  
    }  
  }  
}  
container routing-tables {  
  must "routing-table/name='ipv4-unicast-fib'" {  
    description  
      "IPv4 unicast forwarding information base.";  
  }  
  must "routing-table/name='ipv4-unicast-main'" {  
    description  
      "The main IPv4 unicast routing table.";  
  }  
  description  
    "Container for configured routing tables.";
```



```
list routing-table {
  key "name";
  description
    "Each entry represents a configured routing table. At
    least two entries with names 'ipv4-unicast-fib' and
    'ipv4-unicast-main' must exist.";
  container routes {
    config false;
    description
      "Current contents of the routing table. Note that
      it is operational state data.";
    list route {
      description
        "A routing table entry.";
      leaf destination-prefix {
        type inet:ipv4-prefix;
        description
          "Destination prefix.";
      }
      leaf next-hop {
        type inet:ipv4-address;
        description
          "IPv4 address of the next hop.";
      }
      leaf outgoing-interface {
        type if:interface-ref;
        description
          "Name of the outgoing interface.";
      }
      leaf source-protocol {
        type leafref {
          path "../../../../../routing-protocol-instances/"
            + "routing-protocol-instance/name";
        }
        description
          "Protocol instance from which the route comes.";
      }
      leaf last-modified {
        type yang:date-and-time;
        description
          "Time stamp of the last modification of the
          route. If the route was never modified, it is the
          time when the route was inserted to the routing
          table.";
      }
    }
  }
  leaf name {
```

```
        type string;
        description
            "The name of the routing table.";
    }
    leaf description {
        type string;
        description
            "Textual description of the routing table.";
    }
    list recipient-routing-tables {
        key "recipient-name";
        description
            "A list of routing tables that receive routes from
            the parent routing table.";
        leaf recipient-name {
            type leafref {
                path "../..../routing-table/name";
            }
            description
                "The name of the recipient routing table.";
        }
        leaf filter {
            type leafref {
                path "../..../route-filters/route-filter/name";
            }
            description
                "A route filter which is applied to the routes
                passed on to the recipient routing table.";
        }
    }
}
}
}
}
}
```

<CODE ENDS>

## 7. IANA Considerations

This document registers the following two namespace URIs in the IETF XML registry [RFC3688]:

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

Registrant Contact: The IESG.

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

-----  
URI: urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing

Registrant Contact: The IESG.

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

This document registers two YANG modules in the YANG Module Names registry [RFC6020]:

-----  
name: ietf-routing  
namespace: urn:ietf:params:xml:ns:yang:ietf-routing  
prefix: rt  
reference: RFC XXXX  
-----

-----  
name: ietf-ipv4-unicast-routing  
namespace: urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing  
prefix: v4ur  
reference: RFC XXXX  
-----

## 8. Security Considerations

TBD.

## 9. Acknowledgments

The author wishes to thank Juergen Schoenwaelder and Martin Bjorklund for their helpful comments and suggestions.

## 10. References

### 10.1. Normative References

[IANA-AFI]

IANA, "Address Family Numbers.", January 2011.

[IANA-SAFI]

IANA, "Subsequent Address Family Identifiers (SAFI) Parameters.", March 2011.

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

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

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

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

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

[YANG-IF] Bjorklund, M., "A YANG Data Model for Interface Configuration", draft-bjorklund-netmod-interfaces-cfg-00 (work in progress), December 2010.

### 10.2. Informative References

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.

## Appendix A. Example - Adding a New Routing Protocol

This appendix demonstrates how the core routing data model can be extended to support a new routing protocol. Appendix A.1 contains a YANG module which is used for this purpose. It is intended only as an illustration and not as a real definition of a data model for the RIP routing protocol. Also, for the sake of brevity, we do not follow all the guidelines specified in [RFC6087].

Appendix A.2 then contains a complete instance XML document - a reply to the NETCONF <get> message from a server that uses the RIP protocol as well as static routing.

## A.1. Example YANG Module for Routing Information Protocol

```
module example-rip {
  namespace "http://example.com/rip";
  prefix rip;

  import ietf-interfaces {
    prefix if;
  }
  import ietf-routing {
    prefix rt;
  }

  identity rip {
    base rt:routing-protocol;
    description
      "Identity for the RIP routing protocol.";
  }

  typedef rip-metric {
    type uint8 {
      range "0..16";
    }
  }

  augment "/rt:routing/rt:routing-protocol-instances/" +
    "rt:routing-protocol-instance" {
    when "rt:type='rip:rip'";
    container rip-configuration {
      container rip-interfaces {
        list rip-interface {
          key "name";
          leaf name {
            type if:interface-ref;
          }
        }
      }
    }
  }
}
```

```

        leaf enabled {
            type boolean;
            default "true";
        }
        leaf metric {
            type rip-metric;
            default "1";
        }
        /* Additional per-interface RIP configuration */
    }
}
leaf update-interval {
    type uint8 {
        range "10..60";
    }
    units "seconds";
    default "30";
    description
        "Time interval between periodic updates.";
}
/* Additional RIP configuration */
}
}
augment "/rt:routing/rt:routing-tables/rt:routing-table/rt:route" {
    when "../../../rt:routing-protocol-instances/" +
        "rt:routing-protocol-instance[rt:name=" +
        "current()/rt:source-protocol]/rt:type='rip:rip'";
    description
        "RIP-specific route components.";
    leaf metric {
        type rip-metric;
    }
    leaf tag {
        type uint16;
        default "0";
        description
            "This leaf may be used to carry additional info, e.g. AS
            number.";
    }
}
}
}

```

#### A.2. Sample Reply to the NETCONF <get> Message

This section contains a sample reply to the NETCONF <get> message, which could be sent by a server supporting (and advertizing in <hello>) the following YANG modules:



- o ietf-interfaces [YANG-IF],
- o ex-ethernet [YANG-IF],
- o ex-ip [YANG-IF],
- o ietf-routing (Section 5),
- o ietf-ipv4-unicast-routing (Section 6),
- o example-rip (Appendix A.1).

We assume a simple network setup as shown in Figure 2: routers "ISP" and "A" use RIP for exchanging routing information whereas static routing is used in the private network. In order to avoid the redistribution of the routes to the private subnetworks 192.168.1.0/24 and 192.168.2.0/24 in RIP, an export filter is used in the RIP protocol configuration preventing the routes from the main routing table from appearing in RIP updates.

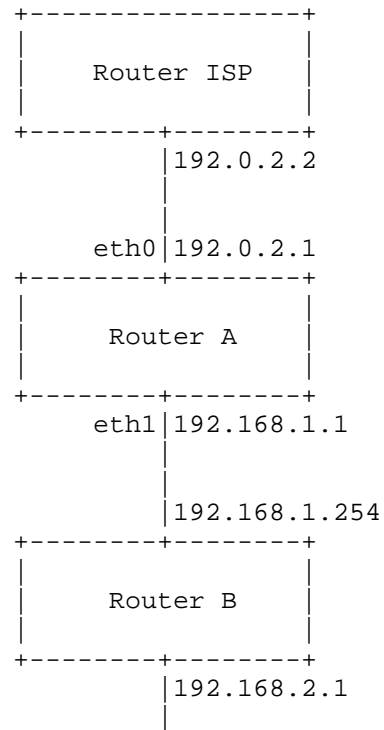


Figure 2: Example network configuration

Router "A" then could send the following XML document as its reply to the NETCONF <get> message:

```
<?xml version="1.0"?>

<nc:rpc-reply
  message-id="101"
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:eth="http://example.com/ethernet"
  xmlns:ip="http://example.com/ip"
  xmlns:rt="urn:ietf:params:xml:ns:yang:ietf-routing"
  xmlns:rip="http://example.com/rip">
  <nc:data>
    <if:interfaces>
      <if:interface>
        <if:name>eth0</if:name>
        <if:type>ethernetCsmacd</if:type>
        <if:location>05:00.0</if:location>
        <ip:ip>
          <ip:address>
            <ip:ip>192.0.2.1</ip:ip>
            <ip:prefix-length>24</ip:prefix-length>
          </ip:address>
        </ip:ip>
      </if:interface>
      <if:interface>
        <if:name>eth1</if:name>
        <if:type>ethernetCsmacd</if:type>
        <if:location>05:00.1</if:location>
        <ip:ip>
          <ip:address>
            <ip:ip>192.168.1.1</ip:ip>
            <ip:prefix-length>24</ip:prefix-length>
          </ip:address>
        </ip:ip>
      </if:interface>
    </if:interfaces>
    <rt:routing>
      <rt:routing-process>
        <rt:name>inet-0</rt:name>
        <rt:address-family>ipV4</rt:address-family>
        <rt:safi>nlri-unicast</rt:safi>
        <ipv4-unicast-routing>
          <routing-protocol-instances>
```

```
<routing-protocol-instance>
  <name>direct</name>
  <type>rt:direct</type>
</routing-protocol-instance>
<routing-protocol-instance>
  <name>st0</name>
  <description>
    Static routing is used for the internal network.
  </description>
  <type>rt:static</type>
  <static-routes>
    <static-route>
      <id>id-6378</id>
      <destination-prefix>192.168.2.0/24</destination-prefix>
      <next-hop>192.168.1.254</next-hop>
    </static-route>
  </static-routes>
</routing-protocol-instance>
<routing-protocol-instance>
  <name>rip0</name>
  <description>
    RIP is used on the uplink.
    Static routes to the internal networks are not
    advertized in RIP.
  </description>
  <type>rip:rip</type>
  <export-filter>deny-all</export-filter>
  <rip:rip-configuration>
    <rip:rip-interfaces>
      <rip:rip-interface>
        <rip:name>eth0</rip:name>
      </rip:rip-interface>
    </rip:rip-interfaces>
  </rip:rip-configuration>
</routing-protocol-instance>
</routing-protocol-instances>
<route-filters>
  <route-filter>
    <name>deny-all</name>
  </route-filter>
</route-filters>
<routing-tables>
  <routing-table>
    <name>ipv4-unicast-fib</name>
    <routes>
      <route>
        <destination-prefix>192.0.2.1/24</destination-prefix>
        <source-protocol>direct</source-protocol>
```

```
    <outgoing-interface>eth0</outgoing-interface>
    <last-modified>2010-04-01T17:11:27+01:00</last-modified>
  </route>
  <route>
    <destination-prefix>192.168.1.0/24</destination-prefix>
    <source-protocol>direct</source-protocol>
    <outgoing-interface>eth1</outgoing-interface>
    <last-modified>2010-04-01T17:11:27+01:00</last-modified>
  </route>
  <route>
    <destination-prefix>192.168.2.0/24</destination-prefix>
    <source-protocol>st0</source-protocol>
    <next-hop>192.168.1.254</next-hop>
    <last-modified>2010-04-01T17:11:32+01:00</last-modified>
  </route>
  <route>
    <destination-prefix>0.0.0.0/0</destination-prefix>
    <source-protocol>rip0</source-protocol>
    <next-hop>192.168.1.254</next-hop>
    <rip:metric>2</rip:metric>
    <rip:tag>64500</rip:tag>
    <last-modified>2010-04-01T18:02:45+01:00</last-modified>
  </route>
</routes>
</routing-table>
<routing-table>
  <name>ipv4-unicast-main</name>
  <recipient-routing-tables>
    <recipient-name>ipv4-unicast-fib</recipient-name>
  </recipient-routing-tables>
  <routes>
    <route>
      <destination-prefix>192.0.2.1/24</destination-prefix>
      <source-protocol>direct</source-protocol>
      <outgoing-interface>eth0</outgoing-interface>
      <last-modified>2010-04-01T17:11:27+01:00</last-modified>
    </route>
    <route>
      <destination-prefix>192.168.1.0/24</destination-prefix>
      <source-protocol>direct</source-protocol>
      <outgoing-interface>eth1</outgoing-interface>
      <last-modified>2010-04-01T17:11:27+01:00</last-modified>
    </route>
    <route>
      <destination-prefix>192.168.2.0/24</destination-prefix>
      <source-protocol>st0</source-protocol>
      <next-hop>192.168.1.254</next-hop>
      <last-modified>2010-04-01T17:11:32+01:00</last-modified>
    </route>
  </routes>
</routing-table>
```

```
    </route>
    <route>
      <destination-prefix>0.0.0.0/0</destination-prefix>
      <source-protocol>rip0</source-protocol>
      <next-hop>192.168.1.254</next-hop>
      <rip:metric>2</rip:metric>
      <rip:tag>64500</rip:tag>
      <last-modified>2010-04-01T18:02:45+01:00</last-modified>
    </route>
  </routes>
</routing-table>
</routing-tables>
</ipv4-unicast-routing>
</rt:routing-process>
</rt:routing>
</nc:data>
</nc:rpc-reply>
```

Author's Address

Ladislav Lhotka  
CESNET

Email: [lhotka@cesnet.cz](mailto:lhotka@cesnet.cz)



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 2, 2012

J. Schoenwaelder  
Jacobs University  
July 1, 2011

Translation of SMIV2 MIB Modules to YANG Modules  
draft-ietf-netmod-smi-yang-01

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.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 2, 2012.

Copyright Notice

Copyright (c) 2011 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 . . . . .	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 . . . . .	10
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 . . . . .	14
8. Translation of the OBJECT-TYPE Macro . . . . .	15
8.1. Scalar and Columnar Object Translation Rules . . . . .	15
8.2. Example: ifNumber and ifIndex of the IF-MIB . . . . .	16
8.3. Non-Augmenting Conceptual Table Translation Rules . . . . .	16
8.4. Example: ifTable of the IF-MIB . . . . .	18
8.5. Example: ifRcvAddressTable of the IF-MIB . . . . .	18
8.6. Augmenting Conceptual Tables Translation Rules . . . . .	20
8.7. Example: ifXTable of the IF-MIB . . . . .	21
9. Translation of the OBJECT-IDENTITY Macro . . . . .	22
9.1. OBJECT-IDENTITY Translation Rules . . . . .	22
9.2. Example: diffServTbParamSimpleTokenBucket of the DIFFSERV-MIB . . . . .	22
10. Translation of the NOTIFICATION-TYPE Macro . . . . .	23
10.1. NOTIFICATION-TYPE Translation Rules . . . . .	23
10.2. Example: linkDown NOTIFICATION-TYPE of IF-MIB . . . . .	23
11. YANG Language Extension Definition . . . . .	26
12. IANA Considerations . . . . .	29
13. Security Considerations . . . . .	30
14. References . . . . .	31
14.1. Normative References . . . . .	31
14.2. Informative References . . . . .	31
Author's Address . . . . .	32

## 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. Implementations are encouraged to provide options to handle situations where DISPLAY-HINTs are added during a revision of a module and backwards compatibility must be preserved.

Mapping of SMIV2 types to YANG types

SMIV2 Module	SMIV2 Type	YANG Module	YANG Type
SNMPv2-SMI	INTEGER	ietf-yang-types	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		ipv4-address
SNMPv2-SMI	Counter32		counter32
SNMPv2-SMI	Gauge32		gauge32
SNMPv2-SMI	TimeTicks		timeticks
SNMPv2-SMI	Opaque		binary
SNMPv2-SMI	Counter64		counter64
SNMPv2-SMI	Unsigned32		uint32
SNMPv2-TC	PhysAddress		phys-address
SNMPv2-TC	MacAddress		mac-address
SNMPv2-TC	TimeStamp		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

- o First, some predefined 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.
- o Otherwise, prefixes are generated by tokenizing an SMIV2 module name, using hyphens as token separators. The tokens derived from 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 SMIPv2 Modules and SMIPv2 IMPORT Clauses

SMIPv2 modules are mapped to corresponding YANG modules. The YANG module name **MUST** be the same as the SMIPv2 module name.

The YANG namespace **MUST** be constructed out of a constant prefix, which followed by the SMIPv2 module name. Since SMIPv2 module names can be assumed to be unique (see Section 3 in [RFC2578]), the resulting YANG namespace is unique. The registered prefix is `urn:ietf:params:xml:ns:yang:smiv2:`, see the IANA considerations in Section 12.

The YANG prefix **MAY** be derived from the SMIPv2 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.

SMIPv2 IMPORT clauses are translated to YANG import statements. One major difference between the SMIPv2 import mechanism and the YANG import mechanism is that SMIPv2 IMPORT clauses import specific symbols from an SMIPv2 module while the YANG import statement imports all symbols of the referenced YANG module.

SMIPv2 imports that are ignored in YANG

SMIPv2 Module	SMIPv2 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-MIB	snmpTraps
SNMPv2-SMI	* all symbols *
SNMPv2-CONF	* all symbols *

Table 3

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

- o 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

defined in this document.

- o 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 in order to produce the imports.

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 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 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 "smiv2-tc"; }  
    import ietf-yang-types     { prefix "yang"; }  
    import ietf-yang-smiv2     { prefix "smiv2"; }  
}
```

## 5. Translation of the MODULE-IDENTITY Macro

The 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.

### 5.1. MODULE-IDENTITY Translation Rules

- o The SMIV2 ORGANIZATION clause is mapped to the YANG organization statement.
- o The SMIV2 CONTACT-INFO clause is mapped to the YANG contact statement.
- o The SMIV2 DESCRIPTION clause is mapped to the YANG description statement.
- o 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.
- o The SMIV2 LAST-UPDATED clause is ignored if the associated date matches a REVISION clause. Otherwise, an additional revision statement is generated.
- o The name of the MODULE-IDENTITY macro invocation is used to generate a top-level container statement. This container MUST be config false.
- o The object identifier value of the invocation of the SMIV2 MODULE-IDENTITY MAY be translated into an smiv2:oid statement contained in the container representing the MODULE-IDENTITY macro invocation, see the YANG extension defined in Section 11.

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, 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.";
}

container ifMIB {
  config false;
  description
    "[Automatically generated top-level container.>";
  smiv2:oid "1.3.6.1.2.1.31";
}
```

## 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 MUST be translated into YANG typedef statements as detailed below.

### 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:

- o The SMIV2 DISPLAY-HINT clause is used to determine the type mapping of types derived from 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.
- o The SMIV2 DISPLAY-HINT MAY be translated into an smiv2:display-hint statement, see the YANG extension defined in Section 11.
- o 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.
- o The SMIV2 DESCRIPTION clause is mapped to the YANG description statement.
- o The SMIV2 REFERENCE clause is mapped to the YANG reference statement.
- o 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.

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].

### 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'.";
  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";
}
```

### 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 SMIV2 uses OBJECT IDENTIFIER assignments to introduce names for intermediate nodes in the OBJECT IDENTIFIER tree. OBJECT IDENTIFIER assignments are not translated into YANG statements.

## 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 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.

### 8.1. 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. 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 into YANG notification leafs.

All leaf statements for scalar objects are created in the top-level container representing the SMIV2 module, see Section 5.1. The leaf statements representing columnar objects are created in the list representing a conceptual row, see Section 8.3.

- o 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.
- o The SMIV2 UNITS clause is mapped to the YANG units statement.
- o The SMIV2 MAX-ACCESS MAY be translated into an smiv2:max-access statement, see the YANG extension defined in Section 11.
- o 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.
- o The SMIV2 DESCRIPTION clause is mapped to the YANG description statement.

- o The SMIV2 REFERENCE clause is mapped to the YANG reference statement.
- o The value of the SMIV2 OBJECT-TYPE macro invocation MAY be translated into an smiv2:oid statement, see the YANG extension defined in Section 11.

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].

### 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.";
  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";
}
```

### 8.3. 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:

- o The SMIV2 SYNTAX clause is ignored
- o The SMIV2 UNITS clause is ignored.
- o The SMIV2 MAX-ACCESS clause is ignored.
- o 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.
- o The SMIV2 DESCRIPTION clause is mapped to the YANG description statement.
- o The SMIV2 REFERENCE clause is mapped to the YANG reference statement.
- o The value of the SMIV2 OBJECT-TYPE macro invocation MAY be translated into an smiv2:oid statement, see the YANG extension defined in Section 11.

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:

- o The SMIV2 SYNTAX clause is ignored.
- o The SMIV2 UNITS clause is ignored.
- o The SMIV2 MAX-ACCESS clause is ignored.
- o 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.
- o The SMIV2 DESCRIPTION clause is mapped to the YANG description statement.
- o The SMIV2 REFERENCE clause is mapped to the YANG reference statement.
- o The SMIV2 INDEX clause is mapped to the YANG key clause listing the columnar objects forming the key of the YANG list.
- o The value of the SMIV2 OBJECT-TYPE macro invocation MAY be translated into an smiv2:oid statement, see the YANG extension defined in Section 11.



Within the list statement, YANG leaf statements are created for columnar objects as described in Section 8.1. 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.

#### 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 {
  config false;
  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";
    }

    // ...
  }
}
```

#### 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."; 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:ifMIB/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.";
    smiv2:max-access "not-accessible";
    smiv2:oid "1.3.6.1.2.1.31.1.4.1.1";
  }

  // ...
}
```

## 8.6. Augmenting Conceptual Tables Translation Rules

An OBJECT-TYPE macro invocation defining an augmenting conceptual table is not translated to a YANG statement. The name assigned by the OBJECT-TYPE macro invocation to the augmenting conceptual table MAY be captured in a comment. The clauses of the macro are translated as follows:

- o The SMIV2 SYNTAX clause is ignored.
- o The SMIV2 UNITS clause is ignored.
- o The SMIV2 MAX-ACCESS clause is ignored.
- o The SMIV2 STATUS clause is ignored.
- o The SMIV2 DESCRIPTION clause MAY be captured in a comment.
- o The SMIV2 REFERENCE clause MAY be captured in a comment.
- o The value of the SMIV2 OBJECT-TYPE macro invocation MAY be captured in a comment.

An OBJECT-TYPE macro invocation defining a conceptual row augmentation is translated to 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:

- o The SMIV2 SYNTAX clause is ignored.
- o The SMIV2 UNITS clause is ignored.
- o The SMIV2 MAX-ACCESS clause is ignored.
- o 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.
- o The SMIV2 DESCRIPTION clause is mapped to the YANG description statement.
- o The SMIV2 REFERENCE clause is mapped to the YANG reference statement.
- o The value of the SMIV2 OBJECT-TYPE macro invocation MAY be translated into an smiv2:oid statement, see the YANG extension defined in Section 11.

Within the augment statement, YANG leaf statements are created as described in Section 8.1.

#### 8.7. Example: ifXTable of the IF-MIB

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

```
/*
 * ifXTable (1.3.6.1.2.1.31.1.1)
 *
 * 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:ifMIB/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";
  }

  // ...
}
```

## 9. 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.

### 9.1. 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 11 as its base. The clauses of the SMIV2 OBJECT-IDENTITY macro are mapped to YANG statements as follows:

- o 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.
- o The SMIV2 DESCRIPTION clause is mapped to the YANG description statement.
- o The SMIV2 REFERENCE clause is mapped to the YANG reference statement.
- o The value of the SMIV2 OBJECT-IDENTITY macro invocation MAY be translated into an smiv2:oid statement, see the YANG extension defined in Section 11.

### 9.2. Example: diffServTBParamSimpleTokenBucket of the DIFFSERV-MIB

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

```
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";
}
```

## 10. Translation of the NOTIFICATION-TYPE Macro

The 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.

### 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.

- o 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 itemized steps in Section 8.1.
- o 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.
- o The SMIV2 DESCRIPTION clause is mapped to the YANG description statement.
- o The SMIV2 REFERENCE clause is mapped to the YANG reference statement.
- o The value of the SMIV2 NOTIFICATION-TYPE macro invocation MAY be translated into an smiv2:oid statement, see the YANG extension defined in Section 11.

### 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.";
  smiv2:oid "1.3.6.1.6.3.1.1.5.3";

  container object-1 {
    description
      "[Automatically generated container for a notification object.";

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

  container object-2 {
    description
      "[Automatically generated container for a notification object.";

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

  container object-3 {
```

```
description
  "[Automatically generated container for a notification object.];

leaf ifIndex {
  type leafref {
    path "/if-mib:ifMIB/if-mib:ifTable" +
        "/if-mib:ifEntry/if-mib:ifIndex";
  }
  description
    "[Automatically generated leaf for a notification object
    index.>";
}
leaf ifOperStatus {
  type leafref {
    path "/if-mib:ifMIB/if-mib:ifTable" +
        "/if-mib:ifEntry/if-mib:ifOperStatus";
  }
  description
    "[Automatically generated leaf for a notification object.>";
}
}
```



## 11. 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@2011-07-01.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 2011-07-01 {
  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
}

identity object-identity {
  description
    "Base identity for all SMIV2 OBJECT-IDENTITYs.";
}

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)";
}

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
    "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 SMIV2 MIB modules into YANG modules, enabling read-only access to data objects defined in SMIV2 MIB modules via NETCONF. The translation itself has no security impact on the Internet.

Users of translated SMIV2 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 [RFC6241] 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.
- [RFC4181] Heard, C., "Guidelines for Authors and Reviewers of MIB Documents", BCP 111, RFC 4181, September 2005.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Author's Address

Juergen Schoenwaelder  
Jacobs University

Email: [j.schoenwaelder@jacobs-university.de](mailto:j.schoenwaelder@jacobs-university.de)





Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: April 5, 2011

A. Bierman  
Brocade  
October 2, 2010

Guidelines for Authors and Reviewers of YANG Data Model Documents  
draft-ietf-netmod-yang-usage-11

Abstract

This memo provides guidelines for authors and reviewers of standards track specifications containing YANG data model modules. Applicable portions may be used as a basis for reviews of other YANG data model documents. Recommendations and procedures are defined, which are intended to increase interoperability and usability of Network Configuration Protocol (NETCONF) implementations which utilize YANG data model modules.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 5, 2011.

Copyright Notice

Copyright (c) 2010 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.

## Table of Contents

1. Introduction . . . . .	4
2. Terminology . . . . .	5
2.1. Requirements Notation . . . . .	5
2.2. NETCONF Terms . . . . .	5
2.3. YANG Terms . . . . .	5
2.4. Terms . . . . .	6
3. General Documentation Guidelines . . . . .	7
3.1. Module Copyright . . . . .	7
3.2. Narrative Sections . . . . .	8
3.3. Definitions Section . . . . .	8
3.4. Security Considerations Section . . . . .	8
3.5. IANA Considerations Section . . . . .	9
3.5.1. Documents that Create a New Name Space . . . . .	9
3.5.2. Documents that Extend an Existing Name Space . . . . .	9
3.6. Reference Sections . . . . .	10
4. YANG Usage Guidelines . . . . .	11
4.1. Module Naming Conventions . . . . .	11
4.2. Identifiers . . . . .	11
4.3. Defaults . . . . .	11
4.4. Conditional Statements . . . . .	12
4.5. XPath Usage . . . . .	12
4.6. Lifecycle Management . . . . .	13
4.7. Module Header, Meta, and Revision Statements . . . . .	14
4.8. Namespace Assignments . . . . .	15
4.9. Top Level Data Definitions . . . . .	16
4.10. Data Types . . . . .	16
4.11. Reusable Type Definitions . . . . .	17
4.12. Data Definitions . . . . .	18
4.13. Operation Definitions . . . . .	19
4.14. Notification Definitions . . . . .	19
5. IANA Considerations . . . . .	20
6. Security Considerations . . . . .	21
6.1. Security Considerations Section Template . . . . .	21
7. Acknowledgments . . . . .	24
8. References . . . . .	25
8.1. Normative References . . . . .	25
8.2. Informative References . . . . .	25
Appendix A. Module Review Checklist . . . . .	27
Appendix B. YANG Module Template . . . . .	29
Appendix C. Change Log . . . . .	32
C.1. Changes from 10 to 11 . . . . .	32
C.2. Changes from 09 to 10 . . . . .	32

C.3.	Changes from 08 to 09	32
C.4.	Changes from 07 to 08	32
C.5.	Changes from 06 to 07	32
C.6.	Changes from 05 to 06	32
C.7.	Changes from 04 to 05	33
C.8.	Changes from 03 to 04	33
C.9.	Changes from 02 to 03	34
C.10.	Changes from 01 to 02	34
C.11.	Changes from 00 to 01	34
Author's Address		36

## 1. Introduction

The standardization of network configuration interfaces for use with the Network Configuration Protocol (NETCONF) [RFC4741] requires a modular set of data models, which can be reused and extended over time.

This document defines a set of usage guidelines for standards track documents containing YANG [I-D.ietf-netmod-yang] data models. YANG is used to define the data structures, protocol operations, and notification content used within a NETCONF server. A server which supports a particular YANG module will support client NETCONF operation requests, as indicated by the specific content defined in the YANG module.

This document is similar to the SMIV2 usage guidelines specification [RFC4181] in intent and structure. However, since that document was written a decade after SMIV2 modules had been in use, it was published as a 'best current practice' (BCP). This document is not a BCP, but rather an informational reference, intended to promote consistency in documents containing YANG modules.

Many YANG constructs are defined as optional to use, such as the description statement. However, in order to maximize interoperability of NETCONF implementations utilizing YANG data models, it is desirable to define a set of usage guidelines which may require a higher level of compliance than the minimum level defined in the YANG specification.

In addition, YANG allows constructs such as infinite length identifiers and string values, or top-level mandatory nodes, that a compliant server is not required to support. Only constructs which all servers are required to support can be used in IETF YANG modules.

This document defines usage guidelines related to the NETCONF operations layer, and NETCONF content layer, as defined in [RFC4741]. These guidelines are intended to be used by authors and reviewers to improve the readability and interoperability of published YANG data models.

## 2. Terminology

### 2.1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

RFC 2119 language is used here to express the views of the NETMOD working group regarding content for YANG modules. YANG modules complying with this document will treat the RFC 2119 terminology as if it were describing best current practices.

### 2.2. NETCONF Terms

The following terms are defined in [RFC4741] and are not redefined here:

- o capabilities
- o client
- o operation
- o server

### 2.3. YANG Terms

The following terms are defined in [I-D.ietf-netmod-yang] and are not redefined here:

- o data node
- o module
- o namespace
- o submodule
- o version
- o YANG
- o YIN

Note that the term 'module' may be used as a generic term for a YANG module or submodule. When describing properties which are specific to submodules, the term 'submodule' is used instead.

## 2.4. Terms

The following terms are used throughout this document:

published: A stable release of a module or submodule, usually contained in an RFC.

unpublished: An unstable release of a module or submodule, usually contained in an Internet-Draft.

### 3. General Documentation Guidelines

YANG data model modules under review are likely to be contained in Internet-Drafts. All guidelines for Internet-Draft authors MUST be followed. These guidelines are defined in [RFC2223] and updated in [RFC5741]. Additional information is also available online at:

<http://www.rfc-editor.org/rfc-editor/instructions2authors.txt>

The following sections MUST be present in an Internet-Draft containing a module:

- o Narrative sections
- o Definitions section
- o Security Considerations section
- o IANA Considerations section
- o References section

#### 3.1. Module Copyright

The module description statement MUST contain a reference to the latest approved IETF Trust Copyright statement, which is available on-line at:

<http://trustee.ietf.org/license-info/>

Each YANG module or submodule contained within an Internet-Draft or RFC is considered to be a code component. The strings '<CODE BEGINS>' and '<CODE ENDS>' MUST be used to identify each code component.

The '<CODE BEGINS>' tag SHOULD be followed by a string identifying the file name specified in section 5.2 of [I-D.ietf-netmod-yang]. The following example is for the '2010-01-18' revision of the 'ietf-foo' module:

```
<CODE BEGINS> file "ietf-foo@2010-01-18.yang"
module ietf-foo {
    // ...
    revision 2010-01-18 {
        description "Latest revision";
        reference "RFC XXXXX";
    }
    // ...
}
<CODE ENDS>
```

Figure 1

### 3.2. Narrative Sections

The narrative part **MUST** include an overview section that describes the scope and field of application of the module(s) defined by the specification and that specifies the relationship (if any) of these modules to other standards, particularly to standards containing other YANG modules. The narrative part **SHOULD** include one or more sections to briefly describe the structure of the modules defined in the specification.

If the module(s) defined by the specification import definitions from other modules (except for those defined in the YANG [I-D.ietf-netmod-yang] or YANG Types [I-D.ietf-netmod-yang-types] documents), or are always implemented in conjunction with other modules, then those facts **MUST** be noted in the overview section, as **MUST** be noted any special interpretations of definitions in other modules.

### 3.3. Definitions Section

This section contains the module(s) defined by the specification. These modules **MUST** be written using the YANG syntax defined in [I-D.ietf-netmod-yang]. A YIN syntax version of the module **MAY** also be present in the document. There **MAY** also be other types of modules present in the document, such as SMIV2, which are not affected by these guidelines.

See Section 4 for guidelines on YANG usage.

### 3.4. Security Considerations Section

Each specification that defines one or more modules **MUST** contain a section that discusses security considerations relevant to those



modules. This section **MUST** be patterned after the latest approved template (available at <http://www.ops.ietf.org/netconf/yang-security-considerations.txt>).

In particular:

- o Writable data nodes that could be especially disruptive if abused **MUST** be explicitly listed by name and the associated security risks **MUST** be explained.
- o Readable data nodes that contain especially sensitive information or that raise significant privacy concerns **MUST** be explicitly listed by name and the reasons for the sensitivity/privacy concerns **MUST** be explained.
- o Operations (i.e., YANG 'rpc' statements) which are potentially harmful to system behavior or that raise significant privacy concerns **MUST** be explicitly listed by name and the reasons for the sensitivity/privacy concerns **MUST** be explained.

### 3.5. IANA Considerations Section

In order to comply with IESG policy as set forth in <http://www.ietf.org/ID-Checklist.html>, every Internet-Draft that is submitted to the IESG for publication which has action items for IANA **MUST** contain an IANA Considerations section. The requirements for this section vary depending what actions are required of the IANA. If there are no IANA considerations applicable to the document, then the IANA Considerations section is not required. Refer to the guidelines in [RFC5226] for more details.

#### 3.5.1. Documents that Create a New Name Space

If an Internet-Draft defines a new name space that is to be administered by the IANA, then the document **MUST** include an IANA Considerations section, that specifies how the name space is to be administered.

Specifically, if any YANG module namespace statement value contained in the document is not already registered with IANA, then a new YANG Namespace registry entry **MUST** be requested from the IANA. The YANG [I-D.ietf-netmod-yang] specification includes the procedure for this purpose in its IANA Considerations section.

#### 3.5.2. Documents that Extend an Existing Name Space

It is possible to extend an existing namespace using a YANG submodule which belongs to an existing module already administered by IANA. In

this case, the document containing the main module MUST be updated to use the latest revision of the submodule.

### 3.6. Reference Sections

For every import or include statement which appears in a module contained in the specification, which identifies a module in a separate document, a corresponding normative reference to that document MUST appear in the Normative References section. The reference MUST correspond to the specific module version actually used within the specification.

For every normative reference statement which appears in a module contained in the specification, which identifies a separate document, a corresponding normative reference to that document SHOULD appear in the Normative References section. The reference SHOULD correspond to the specific document version actually used within the specification. If the reference statement identifies an informative reference, which identifies a separate document, a corresponding informative reference to that document MAY appear in the Informative References section.

#### 4. YANG Usage Guidelines

In general, modules in IETF standards-track specifications MUST comply with all syntactic and semantic requirements of YANG. [I-D.ietf-netmod-yang]. The guidelines in this section are intended to supplement the YANG specification, which is intended to define a minimum set of conformance requirements.

In order to promote interoperability and establish a set of practices based on previous experience, the following sections establish usage guidelines for specific YANG constructs.

Only guidelines which clarify or restrict the minimum conformance requirements are included here.

##### 4.1. Module Naming Conventions

Modules contained in standards track documents SHOULD be named according to the guidelines in the IANA considerations section of [I-D.ietf-netmod-yang].

A distinctive word or acronym (e.g., protocol name or working group acronym) SHOULD be used in the module name. If new definitions are being defined to extend one or more existing modules, then the same word or acronym should be reused, instead of creating a new one.

All published module names MUST be unique. For a YANG module published in an RFC, this uniqueness is guaranteed by IANA. For unpublished modules, the authors need to check that no other work in progress is using the same module name.

Once a module name is published, it MUST NOT be reused, even if the RFC containing the module is reclassified to 'Historic' status.

##### 4.2. Identifiers

Identifiers for all YANG identifiers in published modules MUST be between 1 and 64 characters in length. These include any construct specified as an 'identifier-arg-str' token in the ABNF in section 12 of [I-D.ietf-netmod-yang].

##### 4.3. Defaults

In general, it is suggested that sub-statements containing very common default values SHOULD NOT be present. The following sub-statements are commonly used with the default value, which would make the module difficult to read if used everywhere they are allowed.

Statement	Default Value
config	true
mandatory	false
max-elements	unbounded
min-elements	0
ordered-by	system
status	current
yin-element	false

#### 4.4. Conditional Statements

A module may be conceptually partitioned in several ways, using the 'if-feature' and/or 'when' statements.

Data model designers need to carefully consider all modularity aspects, including the use of YANG conditional statements.

If a data definition is optional, depending on server support for a NETCONF protocol capability, then a YANG 'feature' statement SHOULD be defined to indicate that the NETCONF capability is supported within the data model.

If any notification data, or any data definition, for a non-configuration data node is not mandatory, then the server may or may not be required to return an instance of this data node. If any conditional requirements exist for returning the data node in a notification payload or retrieval request, they MUST be documented somewhere. For example, a 'when' or 'if-feature' statement could apply to the data node, or the conditional requirements could be explained in a 'description' statement within the data node or one of its ancestors (if any).

#### 4.5. XPath Usage

This section describes guidelines for using the XML Path Language [W3C.REC-xpath-19991116] (XPath) within YANG modules.

The 'attribute' and 'namespace' axes are not supported in YANG, and MAY be empty in a NETCONF server implementation.

The 'position' and 'last' functions SHOULD NOT be used. This applies to implicit use of the 'position' function as well (e.g., '//chapter[42]'). A server is only required to maintain the relative XML document order of all instances of a particular user-ordered list or leaf-list. The 'position' and 'last' functions MAY be used if they are evaluated in a context where the context node is a user-ordered 'list' or 'leaf-list'.

The 'preceding', and 'following' axes SHOULD NOT be used. These constructs rely on XML document order within a NETCONF server configuration database, which may not be supported consistently or produce reliable results across implementations. Predicate expressions based on static node properties (e.g., element name or value, 'ancestor' or 'descendant' axes) SHOULD be used instead. The 'preceding' and 'following' axes MAY be used if document order is not relevant to the outcome of the expression (e.g., check for global uniqueness of a parameter value.)

The 'preceding-sibling' and 'following-sibling' axes SHOULD NOT be used. A server is only required to maintain the relative XML document order of all instances of a particular user-ordered list or leaf-list. The 'preceding-sibling' and 'following-sibling' axes MAY be used if they are evaluated in a context where the context node is a user-ordered 'list' or 'leaf-list'.

Data nodes which use the 'int64' and 'uint64' built-in type SHOULD NOT be used within numeric expressions. There are boundary conditions in which the translation from the YANG 64-bit type to an XPath number can cause incorrect results. Specifically, an XPath 'double' precision floating point number cannot represent very large positive or negative 64-bit numbers because it only provides a total precision of 53 bits. The 'int64' and 'uint64' data types MAY be used in numeric expressions if the value can be represented with no more than 53 bits of precision.

Data modelers need to be careful not to confuse the YANG value space and the XPath value space. The data types are not the same in both, and conversion between YANG and XPath data types SHOULD be considered carefully.

Explicit XPath data type conversions MAY be used (e.g., 'string', 'boolean', or 'number' functions), instead of implicit XPath data type conversions.

#### 4.6. Lifecycle Management

The status statement MUST be present if its value is 'deprecated' or 'obsolete'.

The module or submodule name MUST NOT be changed, once the document containing the module or submodule is published.

The module namespace URI value MUST NOT be changed, once the document containing the module is published.

The revision-date sub-statement within the imports statement SHOULD be present if any groupings are used from the external module.

The revision-date sub-statement within the include statement SHOULD be present if any groupings are used from the external sub-module.

If submodules are used, then the document containing the main module MUST be updated so that the main module revision date is equal or more recent than the revision date of any submodule which is (directly or indirectly) included by the main module.

#### 4.7. Module Header, Meta, and Revision Statements

For published modules, the namespace MUST be a globally unique URI, as defined in [RFC3986]. This value is usually assigned by the IANA.

The organization statement MUST be present. If the module is contained in a document intended for standards-track status, then the organization SHOULD be the IETF working group chartered to write the document.

The contact statement MUST be present. If the module is contained in a document intended for standards-track status, then the working group WEB and mailing information MUST be present, and the main document author or editor contact information SHOULD be present. If additional authors or editors exist, their contact information MAY be present. In addition, the Area Director and other contact information MAY be present.

The description statement MUST be present. The appropriate IETF Trust Copyright text MUST be present, as described in Section 3.1.

If the module relies on information contained in other documents, which are not the same documents implied by the import statements present in the module, then these documents MUST be identified in the reference statement.

A revision statement MUST be present for each published version of the module. The revision statement MUST have a reference substatement. It MUST identify the published document which contains the module. Modules are often extracted from their original documents and it is useful for developers and operators to know how

to find the original source document in a consistent manner. The revision statement MAY have a description substatement.

Each new revision MUST include a revision date which is higher than any other revision date in the module. The revision date does not need to be updated if the module contents do not change in the new document revision.

It is acceptable to reuse the same revision statement within unpublished versions (i.e., Internet-Drafts), but the revision date MUST be updated to a higher value each time the Internet-Draft is re-published.

#### 4.8. Namespace Assignments

It is RECOMMENDED that only valid YANG modules are included in documents, whether they are published yet or not. This allows:

- o the module to compile correctly instead of generating disruptive fatal errors.
- o early implementors to use the modules without picking a random value for the XML namespace.
- o early interoperability testing since independent implementations will use the same XML namespace value.

Until a URI is assigned by the IANA, a proposed namespace URI MUST be provided for the namespace statement in a YANG module. A value SHOULD be selected which is not likely to collide with other YANG namespaces. Standard module names, prefixes, and URI strings already listed in the YANG Module Registry MUST NOT be used.

A standard namespace statement value SHOULD have the following form:

```
<URN prefix string>:<module-name>
```

The following URN prefix string SHOULD be used for published and unpublished YANG modules:

```
urn:ietf:params:xml:ns:yang:
```

The following example URNs would be valid temporary namespace statement values for standards-track modules:

```
urn:ietf:params:xml:ns:yang:ietf-netconf-partial-lock
```

```
urn:ietf:params:xml:ns:yang:ietf-netconf-state
```

```
urn:ietf:params:xml:ns:yang:ietf-netconf
```

Note that a different URN prefix string SHOULD be used for non-standards track modules. The string SHOULD be selected according to the guidelines in [I-D.ietf-netmod-yang].

The following examples of non-standards track modules are only suggestions. There are no guidelines for this type of URN in this document:

```
http://example.com/ns/example-interfaces
```

```
http://example.com/ns/example-system
```

#### 4.9. Top Level Data Definitions

There SHOULD only be one top-level data node defined in each YANG module, if any data nodes are defined at all.

The top-level data organization SHOULD be considered carefully, in advance. Data model designers need to consider how the functionality for a given protocol or protocol family will grow over time.

The names and data organization SHOULD reflect persistent information, such as the name of a protocol. The name of the working group SHOULD NOT be used because this may change over time.

A mandatory database data definition is defined as a node that a client must provide for the database to be valid. The server is not required to provide a value.

Top-level database data definitions MUST NOT be mandatory. If a mandatory node appears at the top-level, it will immediately cause the database to be invalid. This can occur when the server boots or when a module is loaded dynamically at runtime.

#### 4.10. Data Types

Selection of an appropriate data type (i.e., built-in type, existing derived type, or new derived type) is very subjective and therefore few requirements can be specified on that subject.

Data model designers SHOULD use the most appropriate built-in data type for the particular application.

If extensibility of enumerated values is required, then the



'identityref' data type SHOULD be used instead of an enumeration or other built-in type.

For string data types, if a machine-readable pattern can be defined for the desired semantics, then one or more pattern statements SHOULD be present.

For string data types, if the length of the string is required to be bounded in all implementations, then a length statement MUST be present.

For numeric data types, if the values allowed by the intended semantics are different than those allowed by the unbounded intrinsic data type (e.g., 'int32'), then a range statement SHOULD be present.

The signed numeric data types (i.e., 'int8', 'int16', 'int32', and 'int64') SHOULD NOT be used unless negative values are allowed for the desired semantics.

For 'enumeration' or 'bits' data types, the semantics for each 'enum' or 'bit' SHOULD be documented. A separate description statement (within each 'enum' or 'bit' statement) SHOULD be present.

#### 4.11. Reusable Type Definitions

If an appropriate derived type exists in any standard module, such as [I-D.ietf-netmod-yang-types], then it SHOULD be used instead of defining a new derived type.

If an appropriate units identifier can be associated with the desired semantics, then a units statement SHOULD be present.

If an appropriate default value can be associated with the desired semantics, then a default statement SHOULD be present.

If a significant number of derived types are defined, and it is anticipated that these data types will be reused by multiple modules, then these derived types SHOULD be contained in a separate module or submodule, to allow easier reuse without unnecessary coupling.

The description statement MUST be present.

If the type definition semantics are defined in an external document (other than another YANG module indicated by an import statement), then the reference statement MUST be present.

#### 4.12. Data Definitions

The description statement **MUST** be present in the following YANG statements:

- o anyxml
- o augment
- o choice
- o container
- o extension
- o feature
- o grouping
- o identity
- o leaf
- o leaf-list
- o list
- o notification
- o rpc
- o typedef

If the data definition semantics are defined in an external document, (other than another YANG module indicated by an import statement), then a reference statement **MUST** be present.

The 'anyxml' construct may be useful to represent an HTML banner containing markup elements, such as '<b>' and '</b>', and **MAY** be used in such cases . However, this construct **SHOULD NOT** be used if other YANG data node types can be used instead to represent the desired syntax and semantics.

If there are referential integrity constraints associated with the desired semantics that can be represented with XPath, then one or more must statements **SHOULD** be present.

For list and leaf-list data definitions, if the number of possible

instances is required to be bounded for all implementations, then the max-elements statements SHOULD be present.

If any must or when statements are used within the data definition, then the data definition description statement SHOULD describe the purpose of each one.

#### 4.13. Operation Definitions

If the operation semantics are defined in an external document (other than another YANG module indicated by an import statement), then a reference statement MUST be present.

If the operation impacts system behavior in some way, it SHOULD be mentioned in the description statement.

If the operation is potentially harmful to system behavior in some way, it MUST be mentioned in the Security Considerations section of the document.

#### 4.14. Notification Definitions

The description statement MUST be present.

If the notification semantics are defined in an external document (other than another YANG module indicated by an import statement), then a reference statement MUST be present.

## 5. IANA Considerations

This document registers one URI in the IETF XML registry [RFC3688]. The following registration is requested:

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

Registrant Contact: The NETMOD WG of the IETF.

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

This document requests the following assignment in the YANG Module Names Registry for the YANG module template in Appendix B.

Field	Value
name	ietf-template
namespace	urn:ietf:params:xml:ns:yang:ietf-template
prefix	temp
reference	RFCXXXX

## 6. Security Considerations

This document defines documentation guidelines for NETCONF content defined with the YANG data modeling language. The guidelines for how to write a Security Considerations section for a YANG module are defined in the online document

<http://www.ops.ietf.org/netconf/yang-security-considerations.txt>

This document does not introduce any new or increased security risks into the management system.

The following section contains the security considerations template dated 2010-06-16. Be sure to check the WEB page at the URL listed above in case there is a more recent version available.

Each specification that defines one or more YANG modules MUST contain a section that discusses security considerations relevant to those modules. This section MUST be patterned after the latest approved template (available at [ed: URL TBD]).

In particular, writable data nodes that could be especially disruptive if abused MUST be explicitly listed by name and the associated security risks MUST be spelled out.

Similarly, readable data nodes that contain especially sensitive information or that raise significant privacy concerns MUST be explicitly listed by name and the reasons for the sensitivity/privacy concerns MUST be explained.

Further, if new RPC operations have been defined, then the security considerations of each new RPC operation MUST be explained.

### 6.1. Security Considerations Section Template

## X. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC4741]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH [RFC4742].

```
-- if you have any writeable data nodes (those are all the
-- "config true" nodes, and remember, that is the default)
-- describe their specific sensitivity or vulnerability.
```

There are a number of data nodes defined in this YANG module which are writable/creatable/deletable (i.e. config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g. edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

<list subtrees and data nodes and state why they are sensitive>

```
-- for all YANG modules you must evaluate whether any readable data
-- nodes (those are all the "config false" nodes, but also all other
-- nodes, because they can also be read via operations like get or
-- get-config) are sensitive or vulnerable (for instance, if they
-- might reveal customer information or violate personal privacy
-- laws such as those of the European Union if exposed to
-- unauthorized parties)
```

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g. via get, get-config or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

<list subtrees and data nodes and state why they are sensitive>

```
-- if your YANG module has defined any rpc operations
-- describe their specific sensitivity or vulnerability.
```

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

<list RPC operations and state why they are sensitive>

Figure 2

## 7. Acknowledgments

The structure and contents of this document are adapted from Guidelines for MIB Documents [RFC4181], by C. M. Heard.

The working group thanks Martin Bjorklund and Juergen Schoenwaelder for their extensive reviews and contributions to this document.



## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2223] Postel, J. and J. Reynolds, "Instructions to RFC Authors", RFC 2223, October 1997.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.
- [RFC5378] Bradner, S. and J. Contreras, "Rights Contributors Provide to the IETF Trust", BCP 78, RFC 5378, November 2008.
- [RFC5741] Daigle, L., Kolkman, O., and IAB, "RFC Streams, Headers, and Boilerplates", RFC 5741, December 2009.
- [W3C.REC-xpath-19991116]  
DeRose, S. and J. Clark, "XML Path Language (XPath) Version 1.0", World Wide Web Consortium Recommendation REC-xpath-19991116, November 1999, <<http://www.w3.org/TR/1999/REC-xpath-19991116>>.
- [I-D.ietf-netmod-yang]  
Bjorklund, M., "YANG - A data modeling language for the Network Configuration Protocol (NETCONF)", draft-ietf-netmod-yang-13 (work in progress), June 2010.
- [I-D.ietf-netmod-yang-types]  
Schoenwaelder, J., "Common YANG Data Types", draft-ietf-netmod-yang-types-09 (work in progress), April 2010.

### 8.2. Informative References

- [RFC4181] Heard, C., "Guidelines for Authors and Reviewers of MIB Documents", BCP 111, RFC 4181, September 2005.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an

IANA Considerations Section in RFCs", BCP 26, RFC 5226,  
May 2008.

## Appendix A. Module Review Checklist

This section is adapted from RFC 4181.

The purpose of a YANG module review is to review the YANG module both for technical correctness and for adherence to IETF documentation requirements. The following checklist may be helpful when reviewing a draft document:

1. I-D Boilerplate -- verify that the draft contains the required Internet-Draft boilerplate (see <http://www.ietf.org/ietf/lid-guidelines.txt>), including the appropriate statement to permit publication as an RFC, and that I-D boilerplate does not contain references or section numbers.
2. Abstract -- verify that the abstract does not contain references, that it does not have a section number, and that its content follows the guidelines in <http://www.ietf.org/ietf/lid-guidelines.txt>.
3. IETF Trust Copyright -- verify that the draft has the appropriate text regarding the rights that document contributors provide to the IETF Trust [RFC5378]. Some guidelines related to this requirement are described in Section 3.1. The IETF Trust license policy (TLP) can be found at:  
  
<http://trustee.ietf.org/docs/IETF-Trust-License-Policy.pdf>
4. Security Considerations Section -- verify that the draft uses the latest approved template from the OPS area web site (<http://www.ops.ietf.org/netconf/yang-security-considerations.txt>) and that the guidelines therein have been followed.
5. IANA Considerations Section -- this section must always be present. For each module within the document, ensure that the IANA Considerations section contains entries for the following IANA registries:  
  
XML Namespace Registry: Register the YANG module namespace.  
  
YANG Module Registry: Register the YANG module name, prefix, namespace, and RFC number, according to the rules specified in [I-D.ietf-netmod-yang].
6. References -- verify that the references are properly divided between normative and informative references, that RFC 2119 is included as a normative reference if the terminology defined therein is used in the document, that all references required by

the boilerplate are present, that all YANG modules containing imported items are cited as normative references, and that all citations point to the most current RFCs unless there is a valid reason to do otherwise (for example, it is OK to include an informative reference to a previous version of a specification to help explain a feature included for backward compatibility). Be sure citations for all imported modules are present somewhere in the document text (outside the YANG module).

7. Copyright Notices -- verify that the draft contains an abbreviated IETF Trust copyright notice in the description statement of each YANG module or sub-module, and that it contains the full IETF Trust copyright notice at the end of the document. Make sure that the correct year is used in all copyright dates. Use the approved text from the latest Trust Legal Provisions (TLP) document, which can be found at:

<http://trustee.ietf.org/license-info/>

8. Other Issues -- check for any issues mentioned in <http://www.ietf.org/ID-Checklist.html> that are not covered elsewhere.

9. Technical Content -- review the actual technical content for compliance with the guidelines in this document. The use of a YANG module compiler is recommended when checking for syntax errors. A list of freely available tools and other information can be found at:

<http://trac.tools.ietf.org/wg/netconf/trac/wiki>

Checking for correct syntax, however, is only part of the job. It is just as important to actually read the YANG module document from the point of view of a potential implementor. It is particularly important to check that description statements are sufficiently clear and unambiguous to allow interoperable implementations to be created.

## Appendix B. YANG Module Template

```
<CODE BEGINS> file "ietf-template@2010-05-18.yang"

module iETF-template {

    // replace this string with a unique namespace URN value
    namespace
        "urn:ietf:params:xml:ns:yang:ietf-template";

    // replace this string, and try to pick a unique prefix
    prefix "temp";

    // import statements here: e.g.,
    // import iETF-yang-types { prefix yang; }
    // import iETF-inet-types { prefix inet; }

    // identify the IETF working group if applicable
    organization
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

    // update this contact statement with your info
    contact
        "WG Web:    <http://tools.ietf.org/wg/your-wg-name/>
        WG List:    <mailto:your-wg-name@ietf.org>

        WG Chair:   your-WG-chair
                   <mailto:your-WG-chair@example.com>

        Editor:     your-name
                   <mailto:your-email@example.com>";

    // replace the first sentence in this description statement.
    // replace the copyright notice with the most recent
    // version, if it has been updated since the publication
    // of this document
    description
        "This module defines a template for other YANG modules.

        Copyright (c) 2010 IETF Trust and the persons identified as
        the document authors.  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

reference "RFC XXXX";

// RFC Ed.: remove this note

// Note: extracted from draft-ietf-netmod-yang-usage-04.txt

// replace '2010-05-18' with the module publication date

// The format is (year-month-day)

```
revision "2010-05-18" {  
  description  
    "Initial version";  
}
```

// extension statements

// feature statements

// identity statements

// typedef statements

// grouping statements

// data definition statements

// augment statements

// rpc statements

// notification statements

// DO NOT put deviation statements in a published module

}

<CODE ENDS>

Figure 3

## Appendix C. Change Log

## C.1. Changes from 10 to 11

- o Removed Intellectual Property section, since no longer required.
- o Reworded XPath guidelines related to XML document order, 'int64' and 'uint64' data types, and 'anyxml' data nodes.

## C.2. Changes from 09 to 10

- o Added security considerations section template.
- o Added guideline for documenting conditional requirements for non-mandatory non-configuration data nodes.
- o Clarified that revision date update applies to the module contents.

## C.3. Changes from 08 to 09

- o Clarifications and corrections to address Gen-ART review comments.

## C.4. Changes from 07 to 08

- o Corrected YANG security considerations URL.
- o Expanded 'CODE BEGINS' example.
- o Added RPC operations to the security considerations guidelines section.
- o Removed guideline about leading and trailing whitespace.

## C.5. Changes from 06 to 07

- o Corrected title change bug; supposed to be page header instead.
- o Fixed typos added to last revision.
- o Added sentence to checklist to make sure text outside module contains citations for imports.

## C.6. Changes from 05 to 06

- o Several clarifications and corrections, based on the AD review by Dan Romascanu.



## C.7. Changes from 04 to 05

- o Changed 'object' terminology to 'data definition'.
- o Put XPath guidelines in separate section.
- o Clarified XPath usage for XML document order dependencies.
- o Updated <CODE BEGINS> guidelines to current conventions.
- o Added informative reference for IANA considerations guidelines and XML registry.
- o Updated IANA Considerations section to reserve the ietf-template module in the YANG Module Name Registry so it cannot be reused accidentally.
- o Many other clarifications and fixed typos found in WGLC reviews.

## C.8. Changes from 03 to 04

- o Removed figure 1 to reduce duplication, just refer to 4741bis draft.
- o Fixed bugs and typos found in WGLC reviews.
- o Removed some guidelines and referring to YANG draft instead of duplicating YANG rules here.
- o Changed security guidelines so they refer to the IETF Trust TLP instead of MIB-specific references.
- o Change temporary namespace guidelines so the DRAFT-XX and RFC-nnnn suffix strings are not used.
- o Changed some MIB boilerplate so it refers to YANG boilerplate instead.
- o Introduced dangling URL reference to online YANG security guidelines

<http://www.ops.ietf.org/yang-security.html>

[ed.: Text from Bert Wijnen will be completed soon and posted online, and then this URL will be finalized.]

- o Moved reference for identifying the source document inside the each revision statement.

- o Removed guideline about valid XPath since YANG already requires valid XPath.
- o Added guideline that strings should not rely on preservation of leading and trailing whitespace characters.
- o Relaxed some XPath and anyxml guidelines from SHOULD NOT or MUST NOT to MAY use with caution.
- o Updated the TLP text within the example module again.
- o Reversed order of change log so most recent entries are first.

C.9. Changes from 02 to 03

- o Updated figure 1 to align with 4741bis draft.
- o Updated guidelines for import-by-revision and include-by-revision.
- o Added file name to code begins convention in ietf-template module.

C.10. Changes from 01 to 02

- o Updated figure 1 per mailing list comments.
- o Updated suggested organization to include the working group name.
- o Updated ietf-template.yang to use new organization statement value.
- o Updated Code Component requirements as per new TLP.
- o Updated ietf-template.yang to use new Code Component begin and end markers.
- o Updated references to the TLP in a couple sections.
- o Change manager/agent terminology to client/server.

C.11. Changes from 00 to 01

- o Added transport 'TLS' to figure 1.
- o Added note about RFC 2119 terminology.
- o Corrected URL for instructions to authors.

- o Updated namespace procedures section.
- o Updated guidelines on module contact, reference, and organization statements.
- o Added note on use of preceding-sibling and following-sibling axes in XPath expressions.
- o Added section on temporary namespace statement values.
- o Added section on top level database objects.
- o Added ietf-template.yang appendix.

Author's Address

Andy Bierman  
Brocade

Email: [andy.bierman@brocade.com](mailto:andy.bierman@brocade.com)

