

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 22, 2014

A. Bierman
YumaWorks
September 18, 2013

YANG Conformance Specification
draft-bierman-netmod-yang-conformance-01

Abstract

This document describes conformance specification and advertisement mechanisms for NETCONF servers implementing YANG data model modules.

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 22, 2014.

Copyright Notice

Copyright (c) 2013 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
1.1.	Terminology	4
1.1.1.	NETCONF	4
1.1.2.	YANG	4
1.1.3.	Terms	5
2.	Problems With YANG Conformance Mechanisms	7
2.1.	YANG Conformance Specification Issues	7
2.1.1.	Import by Revision is Unusable for Conformance	7
2.1.2.	YANG Conformance Specification is Too Simplistic	8
2.1.3.	YANG Deviation Statements Do Not Help	9
2.1.4.	Single Module Conformance in Not Expressive Enough	9
2.2.	Module Capability Advertisement Issues	10
3.	Solution Overview	11
3.1.	Objectives	11
3.2.	YANG Package	11
3.3.	YANG Package File	12
3.4.	Conformance Profile	12
3.5.	Conformance Profile Capability	12
3.6.	YANG Conformance Examples	13
4.	YANG Conformance Statements	20
4.1.	The package Statement	20
4.1.1.	The package Substatements	20
4.2.	The category Statement	20
4.2.1.	The category Substatements	21
4.3.	The subcategory Statement	21
4.4.	The profile Statement	21
4.4.1.	The profile Substatements	21
4.5.	The include-profile Statement	21
4.6.	The require-module Statement	22
4.6.1.	The require-module Substatements	22
4.7.	The min-revision Statement	22
4.8.	The max-revision Statement	23
4.9.	The require-conformance Statement	23
4.10.	The require-feature Statement	24
4.10.1.	The require-feature Substatements	24
4.10.2.	Usage Example	24
4.11.	The require-object Statement	25
4.11.1.	The require-object Substatements	25
4.11.2.	Usage Example	25
4.12.	The require-package Statement	25
4.12.1.	The require-package Substatements	26
4.12.2.	Usage Example	26
4.13.	The require-profile Statement	26
4.14.	The require-capability Statement	26
4.14.1.	The require-capability Substatements	26
4.14.2.	Usage Example	27

5. Updating a YANG Package	28
6. YANG Package Conformance Advertisement	29
7. YANG Conformance ABNF	30
8. IANA Considerations	34
9. Security Considerations	35
10. Open Issues	36
11. Change Log	37
11.1. 00-01	37
12. Normative References	38
Author's Address	39

1. Introduction

There is a need for standard mechanisms to allow YANG [RFC6020] data model designers to express more precise and robust conformance levels for server implementations of a particular YANG module, or set of YANG modules.

There is also a need for standard mechanisms to allow NETCONF [RFC6241] servers to precisely advertise the conformance level of each YANG module it supports.

This document describes some problems with the current conformance specifications mechanisms in YANG and conformance advertisement mechanisms in NETCONF. Solution proposals are also presented to address these problems.

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. NETCONF

The following terms are defined in [RFC6241]:

- o capability
- o client
- o datastore
- o protocol operation
- o server

1.1.2. YANG

The following terms are defined in [RFC6020]:

- o data node
- o extension
- o feature

- o grouping
- o identity
- o module
- o notification
- o submodule
- o typedef

1.1.3. Terms

The following terms are used within this document:

- o conditional node: An object that has one or more "if-feature" sub-statements associated with it. Note that objects affected by "when" statements are not considered conditional for conformance purposes.
- o conformance profile: A set of requirements that a server must support to comply with a given service level. These requirements can be specified in terms of required YANG modules (possibly with specific YANG features and objects supported), other conformance profiles, and/or NETCONF capability URIs (for server functionality that is not specified in YANG modules).
- o import-by-revision: A YANG import statement that includes a revision-date statement. This specifies the exact revision of the YANG module to import, instead of the server picking the revision to import.
- o module base: There is an implied "base" version of the module, which includes all statements which are not conditional. The module base may be empty, a subset of all statements, or the entire module.
- o object: a conceptual data structure represented by a YANG data, rpc, or notification statement.
- o schema tree: The conceptual tree of all objects derived from the set of all YANG modules supported by the server. This tree only includes conditional nodes if all corresponding if-feature statements are "true". Any deviation statements have also been conceptually applied to the schema tree as well.

- o YANG feature set: The set of all objects from a particular module that contain an if-feature statement that corresponds to a particular YANG feature statement.
- o YANG package: A set of conformance profiles that can be extended over time. Also called "package".

2. Problems With YANG Conformance Mechanisms

This section describes some perceived deficiencies with the current data model conformance specification and server conformance advertisement mechanisms used in NETCONF.

2.1. YANG Conformance Specification Issues

The YANG data modeling language provides many powerful data modeling constructs to allow the automation of network configuration protocol operations. However it does not provide enough control over the precise server conformance levels that a client can expect. This has a negative impact on interoperability.

A YANG module is conceptually divided into the module base and zero or more purely optional YANG feature sets.

This approach does not allow enough flexibility and can become difficult to use as the module size and number of YANG feature statements increases. A set of boolean flags that are logically combined as an "AND" expression is too simplistic a mechanism for expressing the criteria for specifying conditional conformance requirements.

2.1.1. Import by Revision is Unusable for Conformance

YANG provides a mechanism to import an exact revision of an external module in order to freeze conformance requirements for a module. If this is not used then the YANG compiler will most likely use the latest revision of the imported module that happens to be implemented by the server.

If new data nodes, notifications, or protocol operations are added to an imported module over time, then it can appear to a NETCONF client that the new objects are implemented if the imported module is updated but not all the modules that import it. Objects using imported typedefs will change syntax and semantics if the typedef (or any typedef it refines) is changed.

Unless import-by-revision is used everywhere an import is used within the dependency chain, the exact module definition cannot really be frozen for conformance purposes.

A server is not required to support multiple revisions of the same module at the same time. This may be very confusing to the client, and complex to implement as well. Instead, servers usually allow only one revision of each module to be implemented within the system.

If import-by-revision is used, then creating a new revision of the imported module requires that the import statements in all the importing modules be updated to use the new revision date. This requires a revision change, so any module that imports those modules also needs to be updated to specify the new revision date of those importing modules. This ripple effect can cause a lot of modules to be updated. It may not be possible to update a module import date in some cases, if that would incorrectly advertise to the client that new objects were implemented by the server.

2.1.2. YANG Conformance Specification is Too Simplistic

Conformance requirements can change over time. New use cases and new consensus about optionality can occur. A conformance statement for each use-case is needed, not just one or more (implied) conformance statements per module.

There are no mechanisms to clearly specify external module dependencies. There is no way to indicate the exact portions of an imported module which are required to comply with a particular conformance level for the importing module. There is no way to specify that multiple modules are required to provide a high-level service.

It is impossible to predict all valid use cases at design time. Partitioning a module into a base plus purely optional features can only account for the features and use cases known at the time. Future designers cannot alter if-feature statements or add new if-feature statements to an augmented module.

YANG features are purely optional to implement. There is no way to specify that a set of objects are conditionally mandatory, based on some data-model specific criteria. Conditions could be expressed with XPath must or when expressions, but this has to be repeated everywhere it is used and the set of objects with the same conditionally mandatory properties is un-named and hard for the reader to identify.

YANG features are too simplistic. They are good for a small number of use-cases within one module. Once there are lots of features, interactions between features, and refined use-cases, they turn the module into a bowl of boolean spaghetti.

YANG features are not really purely optional in practice. Sometimes they are used to express separate roles or service subsets within the module. It is difficult for the reader to identify the valid combinations of purely optional YANG features that represent high-level roles. The YANG if-feature statements are logically combined

as a boolean "AND" expression and not very flexible. YANG description statements are not really machine parsable, so these high-level roles or service groupings are not easily identifiable.

2.1.3. YANG Deviation Statements Do Not Help

YANG deviations could possibly be used as a low-level conformance solution, but they are undesirable and not used by server vendors. YANG deviation statements provide a fairly comprehensive "patch" mechanism to conceptually alter YANG data definition statements. This alteration, or declaration of non-implementation, describes how a server deviates from the standard data definitions.

These statements are not allowed to appear in standard YANG modules, and it turns out that vendors would rather not specify exactly how their server is non-compliant to a standard YANG module. A vendor would rarely need a deviation statement for their own YANG data modules.

YANG deviation statements are too low-level anyway, even if vendors were willing to use them. They do not fully address the future use-case problem because they can only be used to make specific patches to data statements.

2.1.4. Single Module Conformance is Not Expressive Enough

YANG conformance applies only to one module. There are no mechanisms to precisely identify the conformance relationship between modules. Since YANG is designed to be modular and reusable, it is quite likely that a high-level feature or service will be specified with more than one YANG module.

All the top-level definitions are imported from a module whether the importing module uses all the definitions or not. This is too general from a conformance perspective. Sometimes modules are imported just for typedefs or identities, which are always part of the base.

If a module augments a node in another module, it does not imply that it supports all other objects from that module. YANG conformance does not actually address any relationship between modules. There are no mechanisms to express multi-module conformance requirements.

It is difficult for a client application developer to identify the high level server capabilities from a large set of module capabilities. There are no formal mechanisms to identify the definition of a high-level service across multiple modules.

2.2. Module Capability Advertisement Issues

NETCONF servers advertise the YANG modules they support as <capability> URI strings in the <hello> message. The complete list of modules used by the server needs to be advertised in order for the client application to correctly parse the YANG modules and reproduce the schema tree used by the server. However the client does not really know which modules are advertised for full conformance, and which are advertised for partial conformance (such as importing typedef and identity statements from the module).

3. Solution Overview

3.1. Objectives

The solution in this document attempts to achieve several objectives:

- o Provide simple documentation mechanisms that are readable and easy to understand.
- o Provide simple mechanisms that can scale in usage from one module to thousands of modules.
- o Provide per use-case conformance profiles, which allow multiple conformance levels to be specified for a single module.
- o Provide per use-case conformance profiles, which allow multiple modules to be specified for a single conformance profile.
- o Clarify the usage relationship between an augmented module and the augmenting module.
- o Clarify the usage relationship between modules that represent parts of the same conceptual high-level service.
- o Provide the ability to specify a stable conformance definition that cannot implicitly change if YANG modules are updated.
- o Provide the ability to specify the YANG features that must be supported by a server to meet conformance requirements.

3.2. YANG Package

A YANG package is a conformance definition for zero or more YANG modules and/or NETCONF protocol capabilities. Each package has zero or more conformance profiles that describe the server implementation requirements to conform to a specific profile within a package. A YANG package without any conformance profile statements can be used as a placeholder to reserve the package name, but it cannot be advertised as a YANG package capability.

YANG packages are static representations of YANG conformance, meaning there are no server-dependent variables (e.g, set of purely optional YANG features selected by the server). Instead a conformance profile specifies which YANG features a server needs to support to conform to the profile.

YANG packages are defined using a text file similar to YANG modules. However they are separate from YANG modules, since a package can

require more than one YANG module for conformance.

Unlike YANG modules, YANG package definitions do not represent content that would appear in a protocol message. They represent server conformance requirements and are therefore separate from YANG module definitions.

A YANG package is advertised with a <capability> URI string, similar to a YANG module. A NETCONF server will advertise all its supported package capability statements in the <hello> message it sends to each client.

3.3. YANG Package File

A YANG package file consists of UTF-8 characters. The basic syntax is exactly the same as for YANG modules. Several YANG statements are "imported" from the YANG ABNF, and some new statements are defined. Specifically, YANG package syntax is the same as RFC 6020, sections 6, 6.1, 6.2, and 6.3.

[FIXME: not all namespaces in sec. 6.2.1 are supported and a namespace for YANG package names is not defined there.]

At least one revision statement MUST be present in a YANG package file. A new revision MUST be added each time the YANG package file is published. This requirement is more strict than RFC 6020 to ensure that conformance requirements can be properly identified for each server implementation.

3.4. Conformance Profile

A conformance profile represents a conceptual set of server implementation requirements to meet one use-case or variant of the conceptual service represented by the YANG package.

A conformance profile can overlap or even include other conformance profiles. It is a data-model specific matter what requirements make operational sense.

A server can only conform to one conformance profile within a YANG package. Although profile contents can overlap, only one profile per package can be active on a server at a time.

3.5. Conformance Profile Capability

A new NETCONF capability URI is defined to advertise YANG package conformance. A server will announce conformance for a specific conformance profile for each YANG package it supports. Refer to

Section 6 for details on YANG package conformance advertisement.

3.6. YANG Conformance Examples

In this example, 1 conformance profile called "base" is defined for the YANG package named "ietf-types-pkg".

```
package ietf-types-pkg {  
  
  namespace "urn:ietf:params:xml:ns:yang:ietf-types-pkg";  
  prefix "typespkg";  
  organization  
    "IETF NETMOD (NETCONF Data Modeling Language) Working Group";  
  contact " ** WG Chairs ** ";  
  description  
    "This package defines a conformance profile for the standard  
    typedef statements.";  
  
  revision 2013-09-16 {  
    description "First revision";  
    reference "TBD";  
  }  
  
  category general {  
    subcategory types;  
  }  
  
  profile base {  
    description "Basic requirements for YANG types.";  
  
    require-module ietf-yang-types {  
      min-revision "2013-07-15";  
      require-conformance import;  
      description  
        "Support for YANG types is required.";  
      reference "RFC 6991, section 3.";  
    }  
  
    require-module ietf-inet-types {  
      min-revision "2013-07-15";  
      require-conformance import;  
      description  
        "Support for INET types is required.";  
      reference "RFC 6991, section 4.";  
    }  
  }  
}
```

In this example, 4 different conformance profiles are defined for the YANG package named "ietf-routing-pkg":

- o base: a server that supports the base routing profile. This profile is probably not useful without adding routing protocols, but it is needed for extensibility.
- o ipv4: a server that supports IPv4 routing configuration
- o ipv6: a server that supports IPv6 routing configuration
- o ip: a server that supports IPv4 and IPv6 routing configuration

```
package ietf-routing-pkg {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-routing-pkg";  
    prefix "rtpkg";  
    organization  
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";  
    contact " ** WG Chairs ** ";  
    description  
        "This package defines conformance profiles for IPv4 and IPv6  
        routers.";  
    reference "draft-ietf-netmod-routing-cfg-10.txt";  
  
    revision 2013-09-16 {  
        description "First revision";  
        reference "TBD";  
    }  
  
    category protocols {  
        subcategory routing;  
        subcategory ip;  
    }  
  
    profile base {  
        description "Base module requirements for routing";  
        reference "draft-ietf-netmod-routing-cfg-10.txt";  
  
        require-package ietf-types-pkg;  
  
        require-module ietf-routing {  
            min-revision 2013-07-13;  
            require-conformance full;  
            description "The base routing module is required";  
        }  
    }  
}
```

```
    require-module ietf-interfaces {
      min-revision 2013-07-04;
      require-conformance augment;
      description
        "The interface and interface-state tables are augmented.";
    }
  }

  profile ipv4 {
    description "Base module requirements for routing";
    reference "draft-ietf-netmod-routing-cfg-10.txt";

    include-profile base;

    require-module ietf-ipv4-unicast-routing {
      min-revision 2013-07-13;
      require-conformance full;
    }
  }

  profile ipv6 {
    description "Base module requirements for routing";
    reference "draft-ietf-netmod-routing-cfg-10.txt";

    include-profile base;

    require-module ietf-ipv6-unicast-routing {
      min-revision 2013-07-13;
      require-conformance full;
    }
  }

  profile ip {
    description "Base module requirements for routing";
    reference "draft-ietf-netmod-routing-cfg-10.txt";

    include-profile ipv4;

    include-profile ipv6;
  }
}
```

In this example, 5 different conformance profiles are defined for the YANG package named "ietf-netconf-pkg":

- o core: core NETCONF functionality.
- o running: a server that supports writing directly to the running datastore.
- o startup: a server that supports writing directly to the running datastore and also has a separate startup datastore.
- o candidate:- running: a server that supports writing to the candidate datastore and committing all edits at once to the running datastore.
- o confirmed:- running: a server that supports the candidate conformance profile and also supports the confirmed commit operations.

```
package ietf-netconf-pkg {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-pkg";  
    prefix "ncpkg";  
    organization  
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";  
    contact " ** WG Chairs ** ";  
    description  
        "This package defines some conformance profiles for the  
        NETCONF protocol.";  
  
    revision 2013-09-16 {  
        description "First revision";  
        reference "TBD";  
    }  
  
    category protocols {  
        subcategory netconf;  
    }  
  
    profile core {  
        description  
            "Basic requirements for complete NETCONF servers.";  
  
        require-package ietf-types-pkg;  
  
        require-capability "urn:ietf:params:netconf:base:1.1" {  
            description "NETCONF base protocol is required.";  
            reference "RFC 6241, section 8.1";  
        }  
    }  
}
```



```
require-capability
  "urn:ietf:params:netconf:capability:xpath:1.0" {
    description "XPath filtering is required.";
    reference "RFC 6241, section 8.9";
  }

require-capability
  "urn:ietf:params:netconf:capability:validate:1.1" {
    description "NETCONF :validate capability is required.";
    reference "RFC 6241, section 8.6";
  }

require-capability
  "urn:ietf:params:xml:ns:netconf:partial-lock:1.0" {
    description
      "Partial lock capability is required. The YANG module
       ietf-netconf-partial-lock.yang is non-normative
       so a require-module statement is not used instead.";
    reference "RFC 5717, section 4";
  }

require-capability
  "urn:ietf:params:netconf:capability:with-defaults:1.0" {
    description
      "With defaults capability advertisement is required.";
    reference "RFC 6243, section 4.3";
  }

require-capability
  "urn:ietf:params:netconf:capability:notification:1.0" {
    description
      "Notification delivery support is required.";
    reference "RFC 5277, section 3.1";
  }

require-capability
  "urn:ietf:params:netconf:capability:interleave:1.0" {
    description
      "Interleave of commands is required while notification
       delivery is active .";
    reference "RFC 5277, section 6";
  }

require-module ietf-netconf-with-defaults {
  description
    "Support for <with-defaults> RPC parameter is required.";
  reference "RFC 6243, section 5";
}
```

```
require-module ietf-netconf-acm {
  description
    "Base module implementation of NACM is required.";
  reference "RFC 6536, section 3.5.2";
}

require-module ietf-netconf-monitoring {
  description
    "Implementation of NETCONF monitoring is required.";
  reference "RFC 6022, section 5";
}

require-module ietf-netconf-notifications {
  description
    "Implementation of NETCONF base notifications is
    required.";
  reference "RFC 6470, section 2.2";
}
}

profile running {
  description
    "Basic requirements for a complete NETCONF server
    that supports writing directly to the the running
    datastore.";

  include-profile core;

  require-capability
    "urn:ietf:params:netconf:capability:writable-running:1.0" {
    description
      "NETCONF :writable-running capability is required.";
    reference "RFC 6241, section 8.2";
  }

  require-capability
    "urn:ietf:params:netconf:capability:rollback-on-error:1.0" {
    description
      "NETCONF :rollback-on-error capability is required.";
    reference "RFC 6241, section 8.5";
  }
}

profile startup {
  description
    "Basic requirements for a complete NETCONF server
    that supports writing directly to the the running
    datastore and also have a distinct startup datastore.";
```

```
    include-profile running;

    require-capability
      "urn:ietf:params:netconf:capability:startup:1.0" {
        description
          "NETCONF distinct startup capability is required.";
        reference "RFC 6241, section 8.7";
      }
  }

  profile candidate {
    description
      "Basic requirements for a complete NETCONF server
       that supports the candidate datastore.";

    include-profile core;

    require-capability
      "urn:ietf:params:netconf:capability:candidate:1.0" {
        description "NETCONF :candidate capability is required.";
        reference "RFC 6241, section 8.3";
      }
  }

  profile confirmed {
    description
      "Basic requirements for a complete NETCONF server
       that supports the candidate datastore, and confirmed
       commit functionality.";

    include-profile candidate;

    require-capability
      "urn:ietf:params:netconf:capability:confirmed-commit:1.1" {
        description
          "NETCONF :confirmed-commit capability is required.";
        reference "RFC 6241, section 8.4";
      }
  }
}
```

4. YANG Conformance Statements

4.1. The package Statement

The "package" statement defines the YANG package's name, and contains all YANG package header information and conformance profile statements. The "package" statement's argument is the name of the YANG package, followed by a block of substatements that hold detailed package information. The package name follows the rules for identifiers in RFC 6020, section 6.2.

A YANG package name is defined in the same conceptual namespace as YANG module names. The same rules for selecting non-conflicting names apply as defined in RFC 6020, section 7.1.

An IANA registry for YANG package names will be needed, similar to mechanism described in RFC 6020, section 14.

4.1.1. The package Substatements

Substatement	Reference	Cardinality
category	4.2	0..1
contact	RFC 6020, 7.1.8	0..1
description	RFC 6020, 7.19.3	0..1
namespace	RFC 6020, 7.1.3	1
organization	RFC 6020, 7.1.7	0..1
prefix	RFC 6020, 7.1.4	1
profile	4.4	1..n
reference	RFC 6020, 7.19.4	0..1
revision	RFC 6020, 7.1.9	1..n

4.2. The category Statement

The "category" statement, which is optional, takes as an argument the category identifier that best describes the type of functionality provided by the YANG package.

There are no constraints or guidelines on selection of a classification system at this time. It is expected that the IETF will create a classification system for standard YANG modules.

4.2.1. The category Substatements

Substatement	Reference	Cardinality
subcategory	4.3	0..n

4.3. The subcategory Statement

The "subcategory" statement, which is optional, takes as an argument the sub-category identifier that best describes the type of functionality provided by the YANG package. Zero or more subcategory statements are allowed within each category statement;

4.4. The profile Statement

The "profile" statement is used to define one conformance profile within a YANG package. It takes as an argument the profile name, which is followed by a block of substatements that hold detailed conformance information. The package name follows the rules for identifiers in RFC 6020, section 6.2.

4.4.1. The profile Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
include-profile	4.5	0..n
reference	RFC 6020, 7.19.4	0..1
require-capability	4.14	0..n
require-module	4.6	0..n
require-package	4.12	0..n
status	RFC 6020, 7.19.2	0..1

4.5. The include-profile Statement

The "include-profile" statement is used to combine multiple conformance profiles from the same YANG package. It takes as an argument the name of the conformance profile to include. There are no substatements defined. All of the requirements in the included profile are also required in the profile that contains the include-profile statement.

If any require-module, require-package, and/or require-capability statements overlap due to multiple included profiles, then they are

logically combined such that all requirements from all profiles are included.

A profile **MUST NOT** include itself or any conformance profile that would cause itself to be included via a dependency loop.

4.6. The require-module Statement

The "require-module" statement is used to require support for some or all of the definitions in a specific module. It takes as an argument the name of the module to require, followed by a block of substatements that hold detailed module server support requirements.

A require-module statement **MUST NOT** specify the same module name as another require-module statement in the same profile statement.

Submodules are invisible for conformance purposes, because they are used as an implementation mechanism, and are not directly accessible from an external module.

4.6.1. The require-module Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
max-revision	4.8	0..1
min-revision	4.7	0..1
reference	RFC 6020, 7.19.4	0..1
require-conformance	4.9	0..1
require-feature	4.10	0..n
require-object	4.11	0..n

4.7. The min-revision Statement

The "min-revision" statement is used to specify the minimum acceptable revision date for the required module. It takes as argument a date string in the form "YYYY-MM-DD" where "YYYY" is the year, "MM" is the month, and "DD" is the day.

It is used in combination with the "max-revision" statement to identify a range of release dates that are acceptable for profile conformance.

If the min-revision statement is not present, then there is no minimum revision date in the acceptable range.

If the min-revision statement is present, then only revisions of the required module released on or after that date are acceptable for profile conformance.

If the min-revision statement is present, and the max-revision statement is also present, then the min-revision statement MUST represent a date which is the same as or before the max-revision date.

This statement is needed if the conformance profile relies on definitions that were added to the required module in a particular revision.

4.8. The max-revision Statement

The "max-revision" statement is used to specify the maximum acceptable revision date for the required module. It takes as argument a date string in the form "YYYY-MM-DD" where "YYYY" is the year, "MM" is the month, and "DD" is the day.

It is used in combination with the "min-revision" statement to identify a range of release dates that are acceptable for profile conformance.

If the max-revision statement is not present, then there is no maximum revision date in the acceptable range.

If the max-revision statement is present, then only revisions of the required module released on or before that date are acceptable for profile conformance.

If the max-revision statement is present, and the min-revision statement is also present, then the max-revision statement MUST represent a date which is the same as or after the min-revision date.

This statement is needed if the conformance profile relies on definitions that have been changed to obsolete status in the required module, or have been extended or altered in a manner that is not required in the conformance profile.

4.9. The require-conformance Statement

The "require-conformance" statement is used to describe the type of module conformance that is needed to meet the conformance profile requirements. Its argument is an enumerated string value indicating the type of module conformance required.

There are 4 types of module conformance supported:

- o full: Full implementation of the module base is required. This is the default value if the require-conformance statement is not present.
- o augment: full implementation of the objects that are augmented in this module (from the augmenting module) is required.
- o import: the module is required for meta-data definitions, which includes extension, typedef, grouping, identity, and feature statements. No implementation of any objects in the module is required.
- o ad-hoc: Partial implementation of the module base and/or some conditional nodes is required. The require-module statement SHOULD contain "require-object" statements to identify the ad-hoc requirements.

4.10. The require-feature Statement

The "require-feature" statement is used to indicate that the specified YANG feature set is required for profile conformance. It takes as argument the name of the YANG feature that is required, and is followed by a block of substatements that describe the YANG feature usage within the conformance profile.

4.10.1. The require-feature Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
reference	RFC 6020, 7.19.4	0..1

4.10.2. Usage Example

```

require-module ietf-ip {
  min-revision 2013-02-11;
  require-conformance full;
  require-feature ipv4-non-contiguous-netmasks {
    description
      "Configuration of non-contiguous subnet masks
       is required.";
    reference
      "RFC XXXX; Section XXXX";
  }
}

```


4.11. The require-object Statement

The "require-object" statement is used to indicate that the specified YANG object is required for profile conformance. It takes as argument the path string identifying the object. This is similar to a YANG "absolute-schema-nodeid" except that prefixes are not allowed. Only objects defined in the required module can be specified with this statement.

[FIXME: there is no way to specify that the objects that 1 module adds to another with augment-stmt are required in ad-hoc mode.]

4.11.1. The require-object Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
reference	RFC 6020, 7.19.4	0..1

4.11.2. Usage Example

```

require-module ietf-system {
  require-conformance ad-hoc;
  require-object /system-state/clock/current-datetime {
    description
      "The current system time must be provided.";
    reference
      "RFC XXXX; Section XXXX";
  }
}

```

4.12. The require-package Statement

The "require-package" statement is used to require support for an external YANG package. It takes as an argument the name of the YANG package to require, followed by a block of substatements that hold detailed server support requirements.

A require-package statement MUST NOT specify the same YANG package name as another require-package statement in the same profile statement.

4.12.1. The require-package Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
max-revision	4.8	0..1
min-revision	4.7	0..1
reference	RFC 6020, 7.19.4	0..1
require-profile	4.13	0..1

4.12.2. Usage Example

```

require-package ietf-routing-pkg {
  min-revision 2013-09-16;
  require-profile ipv4;
  description
    "Support for IPv4 routing configuration is required.";
}

```

4.13. The require-profile Statement

The "require-profile" statement is used to require support for a specific conformance profile within an external YANG package. It takes as an argument the name of the conformance profile to require.

4.14. The require-capability Statement

The "require-capability" statement is used to indicate that the specified NETCONF capability URI is required for profile conformance. It takes as argument a URI string identifying the NETCONF capability that is required, and is followed by a block of substatements that describe the NETCONF capability usage within the conformance profile.

4.14.1. The require-capability Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
reference	RFC 6020, 7.19.4	0..1

4.14.2. Usage Example

```
profile full-notifications {
  description
    "A profile for requiring full standard NETCONF
    notification functionality.";
  require-capability
    "urn:ietf:params:netconf:capability:notification:1.0" {
      description
        "Support for NETCONF notifications is required.";
      reference "RFC 5277, section 3.1.1";
    }
  require-capability
    "urn:ietf:params:netconf:capability:interleave:1.0" {
      description
        "Support for the ability to accept <rpc> requests when
        NETCONF notification delivery is active is required.";
      reference "RFC 5277, section 6.3";
    }
}
```

5. Updating a YANG Package

A YANG conformance profile definition needs to be altered very carefully after it has been published, in order not to break old clients that expect certain server behavior.

When a new revision of a YANG package is published, the following restrictions apply:

- o An existing conformance profile definition MAY be altered to correct errors in the definition.
- o New statements (e.g. new requirements) MAY be added to an existing conformance profile.
- o Existing requirements MUST NOT be removed from a conformance profile.
- o Existing requirements MUST NOT be altered such that the existing functionality would be taken away from clients.
- o Existing requirements MAY be altered such that the existing functionality appears unaffected to existing clients that are using a previous revision of the conformance profile.
- o An existing conformance profile can be split into multiple new conformance profiles, if the existing conformance profile adds "include-profile" statements such that the required functionality for any existing conformance profile does not change.

6. YANG Package Conformance Advertisement

The YANG Package Conformance capability is used to allow the client to quickly identify which packages and conformance profiles are supported by a particular NETCONF server. The server will advertise each supported YANG package, similar to the YANG module conformance advertisement in RFC 6020, section 5.6.4.

The YANG package namespace URI MUST be advertised as a capability in the NETCONF <hello> message to indicate support for a specific conformance profile within the YANG package. The capability URI MUST be of the form:

```
conf-capability-string  = namespace-uri [ parameter-list ]
parameter-list          = "?" parameter *( "&" parameter )
parameter               = package-parameter /
                           revision-parameter /
                           profile-parameter
package-parameter       = "package=" package-name
revision-parameter      = "revision=" revision-date
profile-parameter       = "profile=" profile-name
```

Where:

- o "package-name" is the name of the YANG package
- o "revision-date" is the revision date of the YANG package
- o "profile-name" is the name of a conformance profile within the YANG package

All 3 parameters MUST be present in the capability string. Refer to Section 4.1 for details on the acceptable values for these parameters.

Example: (capability string wrapped for display purposes only)

```
<capability>urn:ietf:params:xml:ns:yang:ietf-routing-pkg?
  package=ietf-routing-pkg&revision=2013-09-16&profile=ipv4
</capability>
```

7. YANG Conformance ABNF

<CODE BEGINS> file "yang-conformance.abnf"

```
package-stmt      = optsep package-keyword sep identifier-arg-str
                    optsep
                    "{" stmtsep
                      module-header-stmts
                      meta-stmts
                      revision-stmts
                      package-classify-stmts
                      package-body-stmts
                    "}" optsep
```

```
package-classify-stmts = [category-stmt]
```

```
category-stmt = category-keyword sep identifier-arg-str optsep
               (";" /
               "{" stmtsep
                 *(subcategory-stmt stmtsep)
               ")
```

```
subcategory-stmt = subcategory-keyword sep string stmtend
```

```
package-body-stmts = *(profile-stmt stmtsep)
```

```
profile-stmt =    profile-keyword sep identifier-arg-str optsep
                  (";" /
                  "{" stmtsep
                    ;; these stmts can appear in any order
                    [status-stmt stmtsep]
                    [description-stmt stmtsep]
                    [reference-stmt stmtsep]
                    *(include-profile-stmt stmtsep)
                    *(require-package-stmt stmtsep)
                    *(require-capability-stmt stmtsep)
                    *(require-module-stmt stmtsep)
                  "}")
```

```
include-profile-stmt = include-profile-keyword sep
                       identifier-arg-str stmtend
```

```
require-module-stmt = require-module-keyword sep
                      identifier-arg-str optsep
                      (";" /
                      "{" stmtsep
                        ;; these stmts can appear in any order
                        [min-revision-stmt stmtsep]
```

```
        [max-revision-stmt stmtsep]
        [require-conformance-stmt stmtsep]
        [description-stmt stmtsep]
        [reference-stmt stmtsep]
        *(require-feature-stmt stmtsep)
        *(require-object-stmt stmtsep)
    "}")

require-object-stmt = require-object-keyword sep
                    require-object-arg-str optsep
                    (";" /
                    "{" stmtsep
                    ;; these stmts can appear in any order
                    [description-stmt stmtsep]
                    [reference-stmt stmtsep]
                    "}")

require-object-arg-str = < a string that matches the rule
                        require-object-arg >

require-object-arg    = pkg-absolute-schema-nodeid

require-package-stmt = require-package-keyword sep
                    identifier-arg-str optsep
                    (";" /
                    "{" stmtsep
                    ;; these stmts can appear in any order
                    [min-revision-stmt stmtsep]
                    [max-revision-stmt stmtsep]
                    [require-profile-stmt stmtsep]
                    [description-stmt stmtsep]
                    [reference-stmt stmtsep]
                    "}")

require-profile-stmt = require-profile-keyword sep
                    identifier-arg-str stmtend

require-capability-stmt = require-capability-keyword sep
                    uri-str optsep
                    (";" /
                    "{" stmtsep
                    ;; these stmts can appear in any order
                    [description-stmt stmtsep]
                    [reference-stmt stmtsep]
                    "}")

min-revision-stmt = min-revision-keyword sep date-arg-str stmtend
```

```
max-revision-stmt = max-revision-keyword sep date-arg-str stmtend

require-feature-stmt = require-feature-keyword sep
                        identifier-arg-str optsep
                        (";" /
                         "{" stmtsep
                          ;; these stmts can appear in any order
                          [description-stmt stmtsep]
                          [reference-stmt stmtsep]
                         "}")

require-conformance-stmt = require-conformance-keyword sep
                           require-conformance-arg-str stmtend

require-conformance-arg-str =
    < a string that matches the rule
    require-conformance-arg >

require-conformance-arg = full-keyword /
                          augment-keyword /
                          import-keyword /
                          ad-hoc-keyword

pkg-schema-nodeid      = pkg-absolute-schema-nodeid /
                          pkg-descendant-schema-nodeid

pkg-absolute-schema-nodeid = 1*("/") pkg-node-identifier)

pkg-descendant-schema-nodeid =
    pkg-node-identifier
    pkg-absolute-schema-nodeid

pkg-node-identifier = identifier

;; new keywords
ad-hoc-keyword           = 'ad-hoc'
augment-keyword          = 'augment'
category-keyword         = 'category'
conformance-keyword      = 'conformance'
full-keyword             = 'full'
include-profile-keyword  = 'include-profile'
max-revision-keyword     = 'max-revision'
min-revision-keyword     = 'min-revision'
package-keyword          = 'package'
profile-keyword          = 'profile'
require-capability-keyword = 'require-capability'
require-module-keyword   = 'require-module'
require-feature-keyword  = 'require-feature'
```



```
require-package-keyword      = 'require-package'  
require-profile-keyword      = 'require-profile'  
subcategory-keyword          = 'subcategory'
```

```
;; all other symbols are defined in RFC 6020, section 12
```

```
<CODE ENDS>
```

8. IANA Considerations

TBD

9. Security Considerations

TBD

10. Open Issues

- o How can logical OR expressions be supported for modules, features, and capabilities? E.g. a "writable-server" profile will need a require-capability for the :writable-running or for the :candidate capabilities. A server will usually advertise one of these 2 capabilities, but not both of them.
- o Is some sort of "choice-stmt" needed within a profile-stmt, to indicate that 1 of N cases of requirements must be supported? Does the server need to advertise the selected cases in every choice it supports?
- o Can a server support multiple conformance profiles at once? If so, then multiple profiles per package would need to be advertised in the YANG package capability exchange.
- o What if a server is configurable to support different profiles from a given package? Does the server advertise only the configured profile? What if no profile has been configured yet? Is a default conformance profile per YANG package needed?
- o Should Category be removed? If not, how much structure does it need? How many category statements should be allowed per package? Do conformance profiles need category statements?
- o What package and profile naming conventions are needed? Do these identifiers need to share the same namespace as YANG modules?
- o Package file name layout conventions are probably needed like those in RFC 6020, section 5.2. A file extension for a YANG package is needed.
- o Is a media type for a YANG package needed similar to the definition in RFC 6020, section 14.1?
- o Is a statement needed to specify acceptable parameter values for require-capability statements? E.g., a profile that required that the "report-all-tagged" enumeration be supported in the "also-supported" parameter for the ":with-defaults" capability.

11. Change Log

-- RFC Ed.: remove this section before publication.

11.1. 00-01

- o fixed typos in text and examples
- o updated ietf-routing-pkg example
- o added 'require parameters for capabilities' as an open issue

12. 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.
- [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

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Network Working Group
Internet-Draft
Updates: 6020 (if approved)
Intended status: Standards Track
Expires: April 21, 2014

M. Bjorklund
Tail-f Systems
October 18, 2013

YANG XPath Extensions
draft-bjorklund-netmod-yang-xpath-extensions-00

Abstract

This document introduces new YANG extension statements for defining XPath functions. These functions can be used in XPath expressions in YANG modules and in NETCONF XPath filters. A set of YANG-specific XPath functions are also defined.

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 21, 2014.

Copyright Notice

Copyright (c) 2013 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. Defining and Using XPath Functions	4
2.1. Using an XPath Function in YANG	4
2.2. Using an XPath Function in a NETCONF Filter	5
3. XPath Evaluation Context	6
4. XPath Functions	7
4.1. Function for the YANG Types "leafref" and "instance-identifier"	7
4.2. Function for the YANG Type "identityref"	7
4.3. Function for the YANG Type "enumeration"	8
4.4. Function for the YANG Type "bits"	9
4.5. Function for strings	10
5. YANG XPath Extensions Module	11
6. IANA Considerations	17
7. Security Considerations	18
8. References	19
8.1. Normative References	19
8.2. Informative References	19
Author's Address	20

1. Introduction

Experience with YANG [RFC6020] for data modeling has shown that using XPath for specifying constraints is very useful. Unfortunately, since XPath 1.0 has a limited set of data types, and the functions in the core function library only operates on these data types, using XPath 1.0 with other data types is often not possible, unless new XPath functions are defined.

This document defines a mechanism to formally define new XPath functions to be used in YANG modules and NETCONF [RFC6241] XPath filters, and introduces a few such XPath functions to be used for the built-in YANG types that cannot be manipulated efficiently with the core XPath functions.

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. Defining and Using XPath Functions

A YANG extension statement "xpath-function" is introduced in the YANG module in Section 5. It is used to define the name, input parameters, and return type of an XPath function. For example:

```
module example-module1 {
  ...
  import ietf-yang-xpath-extensions {
    prefix yangxp;
  }

  yangxp:xpath-function string-reverse {
    yangxp:xpath-argument str {
      yangxp:xpath-type string;
    }
    yangxp:xpath-result string;
    description
      "This function reverses the string 'str' and returns
       the resulting string.";
  }
}
```

2.1. Using an XPath Function in YANG

When an XPath function defined in a YANG module is used from another module, the module that defines the function is imported, and the function is invoked using the syntax "<prefix>:<function-name>", where <prefix> is the prefix of the imported module. For example:

```
module example-module2 {
  namespace "http://example.com/example-module2";
  ...
  import example-module1 {
    prefix ex1;
  }
  ...
  leaf palindrome-of-the-day {
    type string;
    must ". = ex1:string-reverse(.)" {
      error-message "Not a palindrome.";
    }
  }
}
```

2.2. Using an XPath Function in a NETCONF Filter

An XPath function defined in a YANG module can be used in a NETCONF filter by a client if the NETCONF server advertises the :xpath capability, the capability associated with the YANG module "ietf-yang-xpath-extensions", and the capability associated with the module that defines the XPath function.

For example, suppose a NETCONF server advertises the following capabilities in its <hello> message:

```
<!-- lines wrapped for display purposes only -->

<capability>
  urn:ietf:params:netconf:capability:xpath:1.0
</capability>
<capability>
  urn:ietf:params:xml:ns:yang:ietf-yang-xpath-extensions?
    module=ietf-yang-xpath-extensions
</capability>
<capability>
  http://example.com/example-module2?module=example-module2
</capability>
```

A client can then send the following request to return only interfaces whose names are palindromes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  message-id="1">
  <get xmlns:ex2="http://example.com/example-module2">
    <filter type="xpath"
      select="/interface[name = ex2:reverse(name)]"/>
  </get>
</rpc>
```

3. XPath Evaluation Context

This document updates the XPath evaluation context for YANG XPath expressions, defined in Section 6.4.1 in [RFC6020] in the following way:

- o The function library is the core function library defined in [XPath], and a function "current()" that returns a node set with the initial context node, and all functions defined by the "yangxp:xpath-function" statement in the current module, all included submodules, and all imported modules. Functions defined by "yangxp:xpath-function" are referenced as "<prefix>:<function-name>", where <prefix> is the prefix of the module that defines <function-name>.

4. XPath Functions

This document defines four YANG type-specific XPath functions, and one generic XPath function. The functions are formally defined in Section 5.

4.1. Function for the YANG Types "leafref" and "instance-identifier"

The function "deref" returns a node-set containing the node that a node of type "leafref" or "instance-identifier" refers to. For example:

```
list interface {
  key name;
  leaf name { ... }
  leaf enabled {
    type boolean;
  }
  ...
}

leaf mgmt-interface {
  type leafref {
    path "/interface/name";
  }
  must 'yangxp:deref(..)/../enabled = "true"' {
    error-message
      "The management interface cannot be disabled.";
  }
}
```

4.2. Function for the YANG Type "identityref"

The function "derived-from" checks if a node of type "identityref" is derived from a given identity. For example:

```
module example-interface {  
    ...  
  
    identity interface-type;  
  
    identity ethernet {  
        base interface-type;  
    }  
  
    identity fast-ethernet {  
        base ethernet;  
    }  
  
    identity gigabit-ethernet {  
        base ethernet;  
    }  
  
    list interface {  
        key name;  
        ...  
        leaf type {  
            type identityref {  
                base interface-type;  
            }  
        }  
        ...  
    }  
  
    augment "/interface" {  
        when 'yangxp:derived-from(type,  
                                     "example-interface",  
                                     "ethernet")';  
        // ethernet-specific definitions here  
    }  
}
```

4.3. Function for the YANG Type "enumeration"

The function "enum-value" returns the integer value associated with a node of type "enumeration". For example, with this data model:


```
list alarm {  
  ...  
  leaf severity {  
    type enumeration {  
      enum cleared {  
        value 1;  
      }  
      enum indeterminate {  
        value 2;  
      }  
      enum minor {  
        value 3;  
      }  
      enum warning {  
        value 4;  
      }  
      enum major {  
        value 5;  
      }  
      enum critical {  
        value 6;  
      }  
    }  
  }  
}
```

the following XPath expression selects only alarms that are of severity "major" or higher:

```
/alarm[yangxp:enum-value(severity) >= 5]
```

4.4. Function for the YANG Type "bits"

The function "bit-is-set" checks if a node of type "bits" have a given bit set. For example, if an interface has this leaf:

```
leaf flags {  
  type bits {  
    bit UP;  
    bit PROMISCUOUS;  
    bit DISABLED;  
  }  
}
```

the following XPath expression can be used to select all interfaces with the UP flag set:

```
/interface[bit-is-set(flags, "UP")]
```

4.5. Function for strings

The function "re-match" checks if a string matches a given regular expression. The regular expressions used are the XML Schema regular expressions [XSD-TYPES]. Note that this includes implicit anchoring of the regular expression at the head and tail. For example:

```
re-match('1.22.333', '\d{1,3}\.\d{1,3}\.\d{1,3}')
```

returns true.

To count all logical interfaces called eth0.<number>, do:

```
count(/interface[re-match(name,'eth0\.\d+')])
```

5. YANG XPath Extensions Module

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

```
<CODE BEGINS> file "ietf-yang-xpath-extensions@2013-10-18.yang"

module ietf-yang-xpath-extensions {

    namespace "urn:ietf:params:xml:ns:yang:ietf-yang-xpath-extensions";
    prefix "yangxp";

    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 extensions
        for defining XPath functions to be used in XPath
        expressions in YANG modules and NETCONF filters.

        Copyright (c) 2013 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 2013-10-18 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG XPath Extensions";
}

/*
 * Extensions
 */
```

```
extension xpath-function {
  argument name;
  description
    "This statement introduces an XPath function that can be
    used in 'must' and 'when' XPath expression in YANG modules,
    and in NETCONF filters.
```

The statement's argument specifies the name of the XPath function.

When the function is used in a YANG module, the module where the function is defined MUST be imported. The function is referred to using the syntax '<prefix>:<name>', where <prefix> is the prefix of the module, and <name> is the name of the XPath function.

The following substatements are used:

substatement	cardinality
yangxp:xpath-argument	0..n
yangxp:xpath-result	1
description	0..1
reference	0..1

The yangxp:xpath-argument statement defines the arguments to the XPath function. The functions takes the arguments in the order they are defined in the YANG module."

```
}

extension xpath-argument {
  argument name;
  description
```

"This statement defines an argument to an XPath function.

The statement's argument specifies the name of the XPath function's argument. The mandatory substatement `yangxp:xpath-type` defines the type of the argument.

The following substatements are used:

substatement	cardinality
<code>yangxp:xpath-type</code>	1

```

}

extension xpath-type {
  argument type-name;
  description
    "This statement defines the type of the parent statement's
    XPath object.

    The statement's argument is one of the strings:
    'node-set', 'number', 'string', or 'boolean'.";
}

extension xpath-result {
  argument type;
  description
    "This statement defines the type of the XPath function's
    return value.

    The statement's argument is one of the strings:
    'node-set', 'number', 'string', or 'boolean'.";
}

/*
 * XPath functions
 */

/* Function for leafref and instance-identifier */

yangxp:xpath-function deref {
  yangxp:xpath-argument nodes {
    yangxp:xpath-type node-set;
  }
  yangxp:xpath-result node-set;
  description
    "The deref() function follows the reference defined by the

```

first node in document order in the argument 'nodes', and returns the nodes it refers to.

If the first argument node is of type instance-identifier, the function returns a node-set that contains the single node that the instance identifier refers to, if it exists. If no such node exists, an empty node-set is returned.

If the first argument node is of type leafref, the function returns a node-set that contains the nodes that the leafref refers to.

If the first argument node is of any other type, an empty node-set is returned."

```
reference
"RFC 6020: YANG, Section 9.9 and Section 9.13.";
}

/* Function for identityref */

yangxp:xpath-function derived-from {
  yangxp:xpath-argument nodes {
    yangxp:xpath-type node-set;
  }
  yangxp:xpath-argument module-name {
    yangxp:xpath-type string;
  }
  yangxp:xpath-argument identity-name {
    yangxp:xpath-type string;
  }
  yangxp:xpath-result boolean;
  description
    "The derived-from() function returns true if the first node in
    document order in the argument 'nodes' is a node of type
    identityref, and its value is an identity that is derived from
    the identity 'identity-name' defined in the YANG
    'module-name'; otherwise it returns false.";
  reference
    "RFC 6020: YANG, Section 9.10.";
}

/* Function for enumeration */

yangxp:xpath-function enum-value {
  yangxp:xpath-argument nodes {
    yangxp:xpath-type node-set;
  }
  yangxp:xpath-result number;
```

```
description
  "The enum-value() function checks if the first node in
  document order in the argument 'nodes' is a node of type
  enumeration, and returns the enum's integer value.  If the
  'nodes' node-set is empty, or if the first node 'nodes' is
  not of type enumeration, it returns NaN.";
reference
  "RFC 6020: YANG, Section 9.6.4.2.";
}

/* Function for bits */

yangxp:xpath-function bit-is-set {
  yangxp:xpath-argument nodes {
    yangxp:xpath-type node-set;
  }
  yangxp:xpath-argument bit-name {
    yangxp:xpath-type string;
  }
  yangxp:xpath-result boolean;
description
  "The bit-is-set() function returns true if the first node in
  document order in the argument 'nodes' is a node of type
  bits, and its value has the bit 'bit-name' set; otherwise
  it returns false.";
reference
  "RFC 6020: YANG, Section 9.7.4.";
}

/* String function */

yangxp:xpath-function re-match {
  yangxp:xpath-argument subject {
    yangxp:xpath-type string;
  }
  yangxp:xpath-argument pattern {
    yangxp:xpath-type string;
  }
  yangxp:xpath-result boolean;
description
  "The re-match() function returns true if the 'subject' string
  matches the regular expression 'pattern'; otherwise it
  returns false.

  The regular expressions used are the XML Schema regular
  expressions.";

reference
```

```
        "http://www.w3.org/TR/xmlschema-2/#regexs";  
    }  
}  
  
<CODE ENDS>
```


6. 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-yang-xpath-extensions

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-xpath-extensions
namespace:	urn:ietf:params:xml:ns:yang:ietf-yang-xpath-extensions
prefix:	yangxp
reference:	RFC XXXX

7. Security Considerations

This document defines a formal mechanism for defining XPath functions in YANG data models, and has no security impact on the Internet.

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

8.2. Informative References

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

Author's Address

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: March 26, 2014

A. Clemm
J. Medved
E. Voit
Cisco Systems
September 22, 2013

Mounting YANG-Defined Information from Remote Datastores
draft-clemm-netmod-mount-01.txt

Abstract

This document introduces a new capability that allows YANG datastores to reference and incorporate information from remote datastores. This is accomplished using a new YANG data model that allows to define and manage datastore mount points that reference data nodes in remote datastores. The data model includes a set of YANG extensions for the purposes of declaring such mount points.

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 26, 2014.

Copyright Notice

Copyright (c) 2013 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	2
2. Definitions and Acronyms	5
3. Example scenarios	6
3.1. Network controller view	6
3.2. Distributed network configuration	8
4. Operating on mounted data	9
5. Data model structure	10
5.1. YANG mountpoint extensions	10
5.2. Mountpoint management	11
5.3. YANG structure diagrams	13
5.4. Other considerations	13
5.4.1. Authorization	13
5.4.2. Datastore qualification	14
5.4.3. Local mounting	14
5.4.4. Mount cascades	14
5.4.5. Implementation considerations	15
6. Datastore mountpoint YANG module	16
7. Security Considerations	23
8. Acknowledgements	23
9. References	23
9.1. Normative References	23
9.2. Informative References	23
Appendix A. Example	24

1. Introduction

This document introduces a new capability that allows YANG datastores [RFC6020] to incorporate and reference information from remote datastores. This is provided by introducing a mountpoint concept. This concept allows to declare a YANG data node as a "mount point", under which a remote datastore subtree can be mounted. To the user

of the primary datastore, the remote information appears as an integral part of the datastore. It allows remote data nodes and datastore subtrees to be inserted into the local data hierarchy, arranged below local data nodes. The concept is reminiscent of concepts in a Network File System that allows to mount remote folders and make them appear as if they were contained in the local file system of the user's machine.

The ability to mount information from remote datastores is new and not covered by existing YANG mechanisms. Until now, management information provided in a datastore has been intrinsically tied to the same server. In contrast, the capability introduced here allows the server to represent information from remote systems as if it were its own and contained in its own local data hierarchy.

YANG does provide means by which modules that have been separately defined can reference and augment one another. YANG also does provide means to specify data nodes that reference other data nodes. However, all the data is assumed to be instantiated as part of the same datastore, for example a datastore provided through a NETCONF server [RFC6241]. Existing YANG mechanisms do not account for the possibility that some information that needs to be referred not only resides in a different subtree of the same datastore, or was defined in a separate module that is also instantiated in the same datastore, but that is genuinely part of a different datastore that is provided by a different server.

The ability to mount data from remote datastores is useful to address various problems that several categories of applications are faced with:

One category of applications that can leverage this capability concerns network controller applications that need to present a consolidated view of management information in datastores across a network. Controller applications are faced with the problem that in order to expose information, that information needs to be part of their own datastore. Today, this requires support of a corresponding YANG data module. In order to expose information that concerns other network elements, that information has to be replicated into the controller's own datastore in the form of data nodes that may mirror but are clearly distinct from corresponding data nodes in the network element's datastore. In addition, in many cases, a controller needs to impose its own hierarchy on the data that is different from the one that was defined as part of the original module. An example for this concerns interface configuration data, which would be contained in a top-level container in a network element datastore, but may need to be contained in a list in a controller datastore in order to be able to distinguish instances from different network elements under

the controller's scope. This in turn would require introduction of redundant YANG modules that effectively replicate the same information save for differences in hierarchy.

By directly mounting information from network element datastores, the controller does not need to replicate the same information from multiple datastores, nor does it need to re-define any network element and system-level abstractions to be able to put them in the context of network abstractions. Instead, the subtree of the remote system is attached to the local mount point. Operations that need to access data below the mount point are in effect transparently redirected to remote system, which is the authoritative owner of the data. The mounting system does not even necessarily need to be aware of the specific data in the remote subtree.

A second category of applications concerns decentralized networking applications that require globally consistent configuration of parameters. When each network element maintains its own datastore with the same configurable settings, a single global change requires modifying the same information in many network elements across a network. In case of inconsistent configurations, network failures can result that are difficult to troubleshoot. In many cases, what is more desirable is the ability to configure such settings in a single place, then make them available to every network element. Today, this requires in general the introduction of specialized servers and configuration options outside the scope of NETCONF, such as RADIUS [RFC2866] or DHCP [RFC2131]. In order to address this within the scope of NETCONF and YANG, the same information would have to be redundantly modeled and maintained, representing operational data (mirroring some remote server) on some network elements and configuration data on a designated master. Either way, additional complexity ensues.

Instead of replicating the same global parameters across different datastores, the solution presented in this document allows a single copy to be maintained in a subtree of single datastore that is then mounted by every network element that requires access to these parameters. The global parameters can be hosted in a controller or a designated network element. This considerably simplifies the management of such parameters that need to be known across elements in a network and require global consistency.

The capability of allowing to mount information from remote datastores into another datastore is accomplished by a set of YANG extensions that allow to define such mount points. For this purpose, a new YANG module is introduced. The module defines the YANG extensions, as well as a data model that can be used to manage the mountpoints and mounting process itself. Only the mounting module

and server needs to be aware of the concepts introduced here. Mounting is transparent to the models being mounted; any YANG model can be mounted.

2. Definitions and Acronyms

Data node: An instance of management information in a YANG datastore.

DHCP: Dynamic Host Configuration Protocol.

Datastore: A conceptual store of instantiated management information, with individual data items represented by data nodes which are arranged in hierarchical manner.

Data subtree: An instantiated data node and the data nodes that are hierarchically contained within it.

Mount client: The system at which the mount point resides, into which the remote subtree is mounted.

Mount point: A data node that receives the root node of the remote datastore being mounted.

Mount server: The server with which the mount client communicates and which provides the mount client with access to the mounted information. Can be used synonymously with mount target.

Mount target: A remote server whose datastore is being mounted.

NACM: NETCONF Access Control Model

NETCONF: Network Configuration Protocol

RADIUS: Remote Authentication Dial In User Service.

RPC: Remote Procedure Call

Remote datastore: A datastore residing at a remote node.

URI: Uniform Resource Identifier

YANG: A data definition language for NETCONF

3. Example scenarios

The following example scenarios outline some of the ways in which the ability to mount YANG datastores can be applied. Other mount topologies can be conceived in addition to the ones presented here.

3.1. Network controller view

Network controllers can use the mounting capability to present a consolidated view of management information across the network. This allows network controllers to not only expose network abstractions, such as topologies or paths, but also network element abstractions, such as information about a network element's interfaces, from one consolidated place.

While an application on top of a controller could in theory also bypass the controller to access network elements directly for network-element abstractions, this would come at the expense of added inconvenience for the client application. In addition, it would compromise the ability to provide layered architectures in which access to the network by controller applications is truly channeled through the controller.

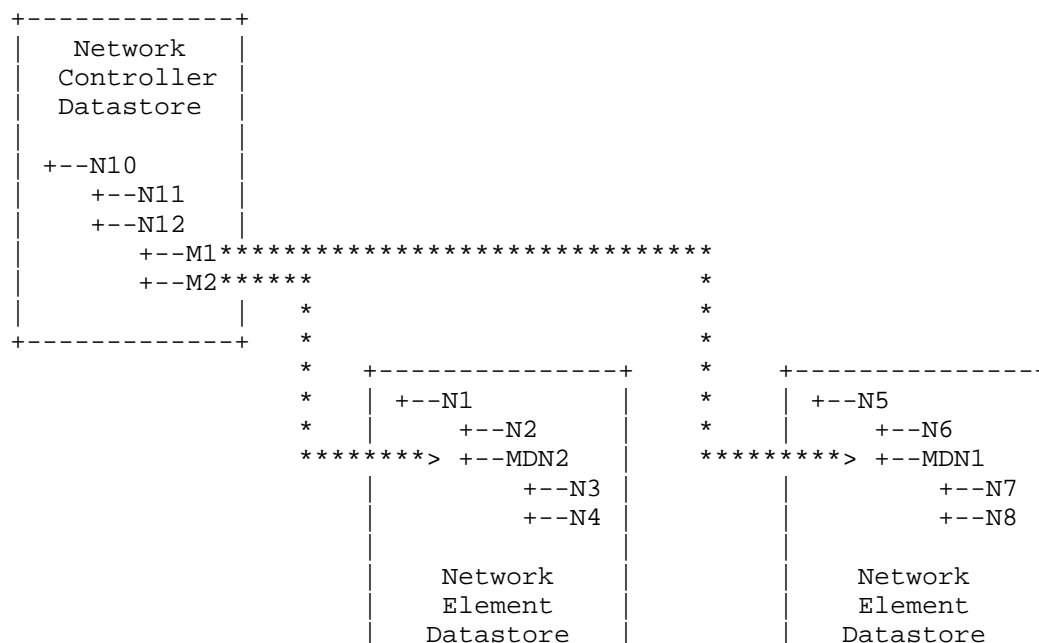
Without a mounting capability, a network controller would need to at least conceptually replicate data from network elements to provide such a view, incorporating network element information into its own controller model that is separate from the network element's, indicating that the information in the controller model is to be populated from network elements. This can introduce issues such as data consistency and staleness. Even more importantly, it would in general lead to the redundant definition of data models: one model that is implemented by the network element itself, and another model to be implemented by the network controller. This leads to poor maintainability, as analogous information has to be redundantly defined and implemented across different data models. In general, controllers cannot simply support the same modules as their network elements for the same information because that information needs to be put into a different context. This leads to "node"-information that needs to be instantiated and indexed differently, because there are multiple instances across different data stores.

For example, "system"-level information of a network element would most naturally be placed into a top-level container at that network element's datastore. At the same time, the same information in the context of the overall network, such as maintained by a controller, might better be provided in a list. For example, the controller might maintain a list with a list element for each network element, underneath which the network element's system-level information is

contained. However, the containment structure of data nodes in a module, once defined, cannot be changed. This means that in the context of a network controller, a second module that repeats the same system-level information would need to be defined, implemented, and maintained. Any augmentations that add additional system-level information to the original module will likewise need to be redundantly defined, once for the "system" module, a second time for the "controller" module.

By allowing a network controller to directly mount information from network element datastores, the controller does not need to replicate the same information from multiple datastores. Perhaps even more importantly, the need to re-define any network element and system-level abstractions to be able to put them in the context of network abstractions is avoided. In this solution, a network controller's datastore mounts information from many network element datastores. For example, the network controller datastore could implement a list in which each list element contains a mountpoint. Each mountpoint mounts a subtree from a different network element's datastore.

This scenario is depicted in Figure 1. In the figure, M1 is the mountpoint for the datastore in Network Element 1 and M2 is the mountpoint for the datastore in Network Element 2. MDN1 is the mounted data node in Network Element 1, and MDN2 is the mounted data node in Network Element 2.



+-----+ +-----+

Figure 1: Network controller mount topology

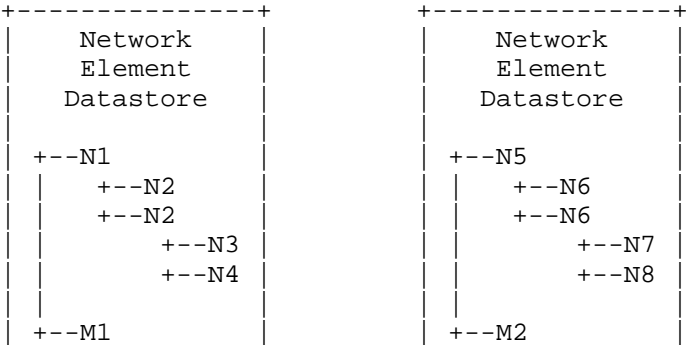
3.2. Distributed network configuration

A second category of applications concerns decentralized networking applications that require globally consistent configuration of parameters that need to be known across elements in a network. Today, the configuration of such parameters is generally performed on a per network element basis, which is not only redundant but, more importantly, error-prone. Inconsistent configurations lead to erroneous network behavior that can be challenging to troubleshoot.

Using the ability to mount information from remote datastores opens up a new possibility for managing such settings. Instead of replicating the same global parameters across different datastores, a single copy is maintained in a subtree of single datastore. This datastore can hosted in a controller or a designated network element. The subtree is subsequently mounted by every network element that requires access to these parameters.

In many ways, this category of applications is an inverse of the previous category: Whereas in the network controller case data from many different datastores would be mounted into the same datastore with multiple mountpoints, in this case many elements, each with their own datastore, mount the same remote datastore, which is then mounted by many different systems.

The scenario is depicted in Figure 2. In the figure, M1 is the mountpoint for the Network Controller datastore in Network Element 1 and M2 is the mountpoint for the Network Controller datastore in Network Element 2. MDN is the mounted data node in the Network Controller datastore that contains the data nodes that represent the shared configuration settings.



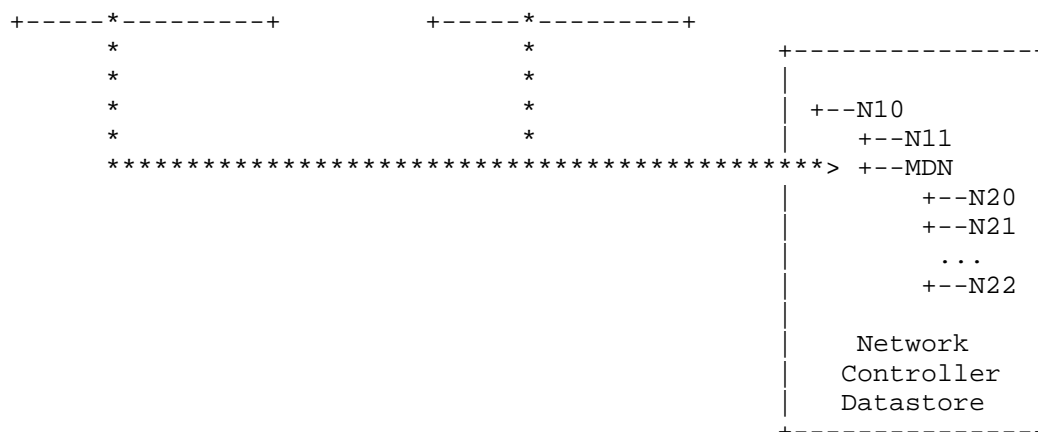


Figure 2: Distributed config settings topology

4. Operating on mounted data

This section provides a rough illustration of the operations flow involving mounted datastores.

The first thing that should be noted about these operations flows concerns that a mount client essentially constitutes a special management application that interacts with a remote system. To the remote system, the mount client constitutes in effect just another application. The remote system is the authoritative owner of the data. While it is conceivable that the remote system (or an application that proxies for the remote system) provides certain functionality to facilitate the specific needs of the mount client, the fact that another system decides to expose a certain "view" of that data is fundamentally not its concern.

When a client makes a request to a server that involves data that is mounted from a remote system, the server will effectively act as a proxy to the remote system on the client's behalf. It will extract from the request the portion that involves the mounted subtree from the remote system. It will strip that portion of the local context, i.e. remove any local data paths and insert the data path of the mounted remote subtree, as appropriate. The server will then forward the transposed request to the remote system that is the authoritative owner of the mounted data. Upon receiving the reply, the server will transpose the results into the local context as needed, for example map the data paths into the local data tree structure, and combine those results with the results of the remainder portion of the original request.

In the simplest and at the same time perhaps the most common case, the request will involve simple data retrieval. In that case, a "get" or "get-configuration" operation might be applied on a subtree whose scope includes a mount point. When resolving the mount point, the server issues its own "get" or "get-configuration" request against the remote system's subtree that is attached to the mount point. The returned information is then inserted into the data structure that is in turn returned to the client that originally invoked the request.

Requests that involve editing of information and "writing through" to remote systems are more complicated, particularly where they involve the need for transactions and locking. While not our primary concern at this time, implications are briefly discussed in section Section 5.4.5.

Since mounted information involves in general communication with a remote system, there is a possibility that the remote system does not respond within a certain amount of time, that connectivity is lost, or that other errors occur. Accordingly, the ability to mount datastores also involves mountpoint management, which includes the ability to configure timeouts, retries, and management of mountpoint state (including dynamic addition removal of mountpoints).

As a final note, it is conceivable that caching schemes are introduced. Caching can increase performance and efficiency in certain scenarios (for example, in the case of data that is frequently read but that rarely changes), but increases implementation complexity. Whether to perform caching is purely a local implementation decision. This specification has not requirement that caching be introduced and makes no corresponding assumptions; there is no dependency on any caching scheme.

5. Data model structure

5.1. YANG mountpoint extensions

At the center of the module is a set of YANG extensions that allow to define a mountpoint.

- o The first extension, "mountpoint", is used to declare a mountpoint. The extension takes the name of the mountpoint as an argument.
- o The second extension, "target", serves as a substatement underneath a mountpoint statement. It takes an argument that identifies the target system. The argument is a reference to a data node that contains the information that is needed to identify

and address a remote server, such as an IP address, a host name, or a URI [RFC3986].

- o The third extension, "subtree", also serves as substatement underneath a mountpoint statement. It takes an argument that defines the root node of the datastore subtree that is to be mounted, specified as string that contains a path expression.

A mountpoint **MUST** be contained underneath a container. Future revisions might allow for mountpoints to be contained underneath other data nodes, such as lists, leaf-lists, and cases. However, to keep things simple, at this point mounting is only allowed directly underneath a container.

Only a single data node can be mounted at one time. While the mount target could refer to any data node, it is recommended that as a best practice, the mount target **SHOULD** refer to a container. It is possibly to maintain e.g. a list of mount points, with each mount point each of which has a mount target an element of a remote list. However, to avoid unnecessary proliferation of the number of mount points and associated management overhead, in order to mount lists or leaf-lists, a container containing the list respectively leaf-list **SHOULD** be mounted.

It is possible for a mounted datastore to contain another mountpoint, thus leading to several levels of mount indirections. However, mountpoints **MUST NOT** introduce circular dependencies. In particular, a mounted datastore **MUST NOT** contain a mountpoint which specifies the mounting datastore as a target and a subtree which contains as root node a data node that in turn contains the original mountpoint. Whenever a mount operation is performed, this condition **MUST** be validated by the mount client.

5.2. Mountpoint management

The YANG module contains facilities to manage the mountpoints themselves.

For this purpose, a list of the mountpoints is introduced. Each list element represents a single mountpoint. It includes an identification of the mount target, i.e. the remote system hosting the remote datastore and a definition of the subtree of the remote data node being mounted. It also includes monitoring information about current status (indicating whether the mount has been successful and is operational, or whether an error condition applies such as the target being unreachable or referring to an invalid subtree).

In addition to the list of mountpoints, a set of global mount policy settings allows to set parameters such as mount retries and timeouts.

Each mountpoint list element also contains a set of the same configuration knobs, allowing administrators to override global mount policies and configure mount policies on a per-mountpoint basis if needed.

There are two ways how mounting occurs: automatic (dynamically performed as part of system operation) or manually (administered by a user or client application). A separate mountpoint-origin object is used to distinguish between manually configured and automatically populated mountpoints.

When configured automatically, mountpoint information is automatically populated by the datastore that implements the mountpoint. The precise mechanisms for discovering mount targets and bootstrapping mount points are provided by the mount client infrastructure and outside the scope of this specification. Likewise, when a mountpoint should be deleted and when it should merely have its mount-status indicate that the target is unreachable is a system-specific implementation decision.

Manual mounting consists of two steps. In a first step, a mountpoint is manually configured by a user or client application through administrative action. Once a mountpoint has been configured, actual mounting occurs through an RPCs that is defined specifically for that purpose. To unmount, a separate RPC is invoked; mountpoint configuration information needs to be explicitly deleted.

The structure of the mountpoint management data model is depicted in the following figure, where brackets enclose list keys, "rw" means configuration, "ro" operational state data, and "?" designates optional nodes. Parentheses enclose choice and case nodes. The figure does not depict all definitions; it is intended to illustrate the overall structure.

```

rw mount-server-mgmt
+-- rw mountpoints
|   +-- rw mountpoint [mountpoint-id]
|       +-- rw mountpoint-id  string
|       +-- rw mount-target
|           +--: (IP)
|               +-- rw target-ip  yang:ip-address
|           +--: (URI)
|               +-- rw uri  yang:uri
|           +--: (host-name)
|               +-- rw hostname  yang:host

```

```

|         | +--- (node-ID)
|         | |   +--- rw node-info-ref  mnt:subtree-ref
|         | +--- (other)
|         | |   +--- rw opaque-target-id  string
|         +--- rw subtree-ref  mnt:subtree-ref
|         +--- ro mountpoint-origin enumeration
|         +--- ro mount-status  mnt:mount-status
|         +--- rw manual-mount? empty
|         +--- rw retry-timer? uint16
|         +--- rw number-of-retries? uint8
+--- rw global-mount-policies
    +--- rw manual-mount? empty
    +--- rw retry-time? uint16
    +--- rw number-of-retries? uint8

```

5.3. YANG structure diagrams

YANG data model structure overviews have proven very useful to convey the "Big Picture". It would be useful to indicate in YANG data model structure overviews the fact that a given data node serves as a mountpoint. We propose for this purpose also a corresponding extension to the structure representation convention. Specifically, we propose to prefix the name of the mounting data node with upper-case 'M'.

```

rw network
+--- rw nodes
    +--- rw node [node-ID]
        +--- rw node-ID
        +--- M node-system-info

```

5.4. Other considerations

5.4.1. Authorization

Whether a mount client is allowed to modify information in a mounted datastore or only retrieve it and whether there are certain data nodes or subtrees within the mounted information for which access is restricted is subject to authorization rules. To the mounted system, a mounting client will in general appear like any other client. Authorization privileges for remote mounting clients need to be specified through NACM (NETCONF Access Control Model) [RFC6536].

Users and implementers need to be aware of certain issues when mounted information is modified, not just retrieved. Specifically, in certain corner cases validation of changes made to mounted data

may involve constraints that involve information that is not visible to the mounting datastore. This means that in such cases the reason for validation failures may not always be fully understood by the mounting system.

Likewise, if the concepts of transactions and locking are applied at the mounting system, these concepts will need to be applied across multiple systems, not just across multiple data nodes within the same system. This capability may not be supported by every implementation. For example, locking a datastore that contains a mountpoint requires that the mount client obtains corresponding locks on the mounted datastore as needed. Any request to acquire a lock on a configuration subtree that includes a mountpoint **MUST NOT** be granted if the mount client fails to obtain a corresponding lock on the mounted system. Likewise, in case transactions are supported by the mounting system, but not the target system, requests to acquire a lock on a configuration subtree that includes a mountpoint **MUST NOT** be granted.

5.4.2. Datastore qualification

It is conceivable to differentiate between different datastores on the remote server, that is, to designate the name of the actual datastore to mount, e.g. "running" or "startup". However, for the purposes of this spec, we assume that the datastore to be mounted is generally implied. Mounted information is treated as analogous to operational data; in general, this means the running or "effective" datastore is the target. That said, the information which targets to mount does constitute configuration and can hence be part of a startup or candidate datastore.

5.4.3. Local mounting

It is conceivable that the mount target does not reside in a remote datastore, but that data nodes in the same datastore as the mountpoint are targeted for mounting. This amounts to introducing an "aliasing" capability in a datastore. While this is not the scenario that is primarily targeted, it is supported and there may be valid use cases for it.

5.4.4. Mount cascades

It is possible for the mounted subtree to in turn contain a mountpoint. However, circular mount relationships MUST NOT be introduced. For this reason, a mounted subtree MUST NOT contain a mountpoint that refers back to the mounting system with a mount target that directly or indirectly contains the originating mountpoint. As part of a mount operation, the mount points of the mounted system need to be checked accordingly.

5.4.5. Implementation considerations

Implementation specifics are outside the scope of this specification. That said, the following considerations apply:

Systems that wish to mount information from remote datastores need to implement a mount client. The mount client communicates with a remote system to access the remote datastore. To do so, there are several options:

- o The mount client acts as a NETCONF client to a remote system. Alternatively, another interface to the remote system can be used, such as a REST API using JSON encodings, as specified in [I-D.bierman-netconf-restconf]. Either way, to the remote system, the mount client constitutes essentially a client application like any other. The mount client in effect IS a special kind of client application.
- o The mount client communicates with a remote mount server through a separate protocol. The mount server is deployed on the same system as the remote NETCONF datastore and interacts with it through a set of local APIs.
- o The mount client communicates with a remote mount server that acts as a NETCONF client proxy to a remote system, on the client's behalf. The communication between mount client and remote mount server might involve a separate protocol, which is translated into NETCONF operations by the remote mount server.

It is the responsibility of the mount client to manage the association with the target system, e.g. validate it is still reachable by maintaining a permanent association, perform reachability checks in case of a connectionless transport, etc.

It is the responsibility of the mount client to manage the mountpoints. This means that the mount client needs to populate the mountpoint monitoring information (e.g. keep mount-status up to data and determine in the case of automatic mounting when to add and remove mountpoint configuration). In the case of automatic mounting, the mount client also interacts with the mountpoint discovery and bootstrap process.

The mount client needs to also participate in servicing datastore operations involving mounted information. An operation requested involving a mountpoint is relayed by the mounting system's infrastructure to the mount client. For example, a request to retrieve information from a datastore leads to an invocation of an internal mount client API when a mount point is reached. The mount client then relays a corresponding operation to the remote datastore. It subsequently relays the result along with any responses back to the invoking infrastructure, which then merges the result (e.g. a retrieved subtree with the rest of the information that was retrieved) as needed. Relaying the result may involve the need to transpose error response codes in certain corner cases, e.g. when mounted information could not be reached due to loss of connectivity with the remote server, or when a configuration request failed due to validation error.

6. Datastore mountpoint YANG module

```
<CODE BEGINS>
file "mount@2013-09-22.yang"
module mount {
  namespace "urn:cisco:params:xml:ns:yang:mount";
  // replace with IANA namespace when assigned

  prefix mnt;

  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: netmod@ietf.org

    WG Chair: David Kessens
    david.kessens@nsn.com
```

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

Editor: Alexander Clemm
alex@cisco.com";

```
description
  "This module provides a set of YANG extensions and definitions
  that can be used to mount information from remote datastores.";
```

```
revision 2013-09-22 {
  description "Initial revision.";
}
```

```
feature mount-server-mgmt {
  description
    "Provide additional capabilities to manage remote mount
    points";
}
```

```
extension mountpoint {
  description
    "This YANG extension is used to mount data from a remote
    system in place of the node under which this YANG extension
    statement is used.
```

This extension takes one argument which specifies the name of the mountpoint.

This extension can occur as a substatement underneath a container statement, a list statement, or a case statement. As a best practice, it SHOULD occur as statement only underneath a container statement, but it MAY also occur underneath a list or a case statement.

The extension takes two parameters, target and subtree, each defined as their own YANG extensions. A mountpoint statement MUST contain a target and a subtree substatement for the mountpoint definition to be valid.

The target system MAY be specified in terms of a data node that uses the grouping 'mnt:mount-target'. However, it can be specified also in terms of any other data node that contains sufficient information to address the mount target, such as an IP address, a host name, or a URI.

The subtree SHOULD be specified in terms of a data node of type 'mnt:subtree-ref'. The targeted data node MUST

represent a container.

It is possible for the mounted subtree to in turn contain a mountpoint. However, circular mount relationships MUST NOT be introduced. For this reason, a mounted subtree MUST NOT contain a mountpoint that refers back to the mounting system with a mount target that directly or indirectly contains the originating mountpoint.";

```
    argument "name";
}

extension target {
  description
    "This YANG extension is used to specify a remote target
    system from which to mount a datastore subtree. This YANG
    extension takes one argument which specifies the remote
    system. In general, this argument will contain the name of
    a data node that contains the remote system information. It
    is recommended that the reference data node uses the
    mount-target grouping that is defined further below in this
    module.

    This YANG extension can occur only as a substatement below
    a mountpoint statement. It MUST NOT occur as a substatement
    below any other YANG statement.";

    argument "target-name";
}

extension subtree {
  description
    "This YANG extension is used to specify a subtree in a
    datastore that is to be mounted. This YANG extension takes
    one argument which specifies the path to the root of the
    subtree. The root of the subtree SHOULD represent an
    instance of a YANG container. However, it MAY represent
    also another data node.

    This YANG extension can occur only as a substatement below
    a mountpoint statement. It MUST NOT occur as a substatement
    below any other YANG statement.";

    argument "subtree-path";
}

typedef mount-status {
  description
```

```
        "This type is used to represent the status of a
        mountpoint.";
    type enumeration {
        enum ok {
            description
                "Mounted";
        }
        enum no-target {
            description
                "The argument of the mountpoint does not define a
                target system";
        }
        enum no-subtree {
            description
                "The argument of the mountpoint does not define a
                root of a subtree";
        }
        enum target-unreachable {
            description
                "The specified target system is currently
                unreachable";
        }
        enum mount-failure {
            description
                "Any other mount failure";
        }
        enum unmounted {
            description
                "The specified mountpoint has been unmounted as the
                result of a management operation";
        }
    }
}

typedef subtree-ref {
    type string; // string pattern to be defined
    description
        "This string specifies a path to a datanode. It corresponds
        to the path substatement of a leafref type statement. Its
        syntax needs to conform to the corresponding subset of the
        XPath abbreviated syntax. Contrary to a leafref type,
        subtree-ref allows to refer to a node in a remote datastore.
        Also, a subtree-ref refers only to a single node, not a list
        of nodes.";
}

rpc mount {
    description
        "This RPC allows an application or administrative user to
        perform a mount operation. If successful, it will result in
```



```
        the creation of a new mountpoint.";
    input {
        leaf mountpoint-id {
            type string {
                length "1..32";
            }
        }
    }
    output {
        leaf mount-status {
            type mount-status;
        }
    }
}
rpc unmount {
    "This RPC allows an application or administrative user to
    unmount information from a remote datastore.  If successful,
    the corresponding mountpoint will be removed from the
    datastore.";
    input {
        leaf mountpoint-id {
            type string {
                length "1..32";
            }
        }
    }
    output {
        leaf mount-status {
            type mount-status;
        }
    }
}
grouping mount-monitor {
    leaf mount-status {
        description
            "Indicates whether a mountpoint has been successfully
            mounted or whether some kind of fault condition is
            present.";
        type mount-status;
        config false;
    }
}
grouping mount-target {
    description
        "This grouping contains data nodes that can be used to
        identify a remote system from which to mount a datastore
        subtree.";
    container mount-target {
```

```
choice target-address-type {
  mandatory;
  case IP {
    leaf target-ip {
      type yang:ip-address;
    }
  }
  case URI {
    leaf uri {
      type yang:uri;
    }
  }
  case host-name {
    leaf hostname {
      type yang:host;
    }
  }
  case node-ID {
    leaf node-info-ref {
      type subtree-ref;
    }
  }
  case other {
    leaf opaque-target-ID {
      type string;
      description
        "Catch-all; could be used also for mounting
        of data nodes that are local.";
    }
  }
}

}

}

}

grouping mount-policies {
  description
    "This grouping contains data nodes that allow to configure
    policies associated with mountpoints.";
  leaf manual-mount {
    type empty;
    description
      "When present, a specified mountpoint is not
      automatically mounted when the mount data node is
      created, but needs to be mounted via specific RPC
      invocation.";
  }
  leaf retry-timer {
    type uint16;
    units "seconds";
    description

```

```
        "When specified, provides the period after which
        mounting will be automatically reattempted in case of a
        mount status of an unreachable target";
    }
    leaf number-of-retries {
        type uint8;
        description
            "When specified, provides a limit for the number of
            times for which retries will be automatically
            attempted";
    }
}

container mount-server-mgmt {
    if-feature mount-server-mgmt;
    container mountpoints {
        list mountpoint {
            key "mountpoint-id";

            leaf mountpoint-id {
                type string {
                    length "1..32";
                }
            }
            leaf mountpoint-origin {
                type enumeration {
                    enum client {
                        description
                            "Mountpoint has been supplied and is
                            manually administered by a client";
                    }
                    enum auto {
                        description
                            "Mountpoint is automatically
                            administered by the server";
                    }
                }
                config false;
            }
        }
        uses mount-target;
        leaf subtree-ref {
            type subtree-ref;
            mandatory;
        }
        uses mount-monitor;
        uses mount-policies;
    }
}
```

```
        container global-mount-policies {
            uses mount-policies;
            description
                "Provides mount policies applicable for all mountpoints,
                unless overridden for a specific mountpoint.";
        }
    }
}
<CODE ENDS>
```

7. Security Considerations

TBD

8. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Tony Tkacik, Robert Varga, Lukas Sedlak, and Benoit Claise.

9. References

9.1. Normative References

- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.
- [RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

9.2. Informative References

[I-D.bierman-netconf-restconf]

Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando,
 "RESTCONF Protocol", draft-bierman-netconf-restconf-01
 (work in progress), September 2013.

Appendix A. Example

In the following example, we are assuming the use case of a network controller that wants to provide a controller network view to its client applications. This view needs to include network abstractions that are maintained by the controller itself, as well as certain information about network devices where the network abstractions tie in with element-specific information. For this purpose, the network controller leverages the mount capability specified in this document and presents a fictitious Controller Network YANG Module that is depicted in the outlined structure below. The example illustrates how mounted information is leveraged by the mounting datastore to provide an additional level of information that ties together network and device abstractions, which could not be provided otherwise without introducing a (redundant) model to replicate those device abstractions

```
rw controller-network
+-- rw topologies
|   +-- rw topology [topo-id]
|       +-- rw topo-id          node-id
|       +-- rw nodes
|           +-- rw node [node-id]
|               +-- rw node-id          node-id
|               +-- rw supporting-ne    network-element-ref
|               +-- rw termination-points
|                   +-- rw term-point [tp-id]
|                       +-- tp-id        tp-id
|                       +-- ifref         mountedIfRef
|       +-- rw links
|           +-- rw link [link-id]
|               +-- rw link-id          link-id
|               +-- rw source            tp-ref
|               +-- rw dest              tp-ref
+-- rw network-elements
    +-- rw network-element [element-id]
        +-- rw element-id              element-id
        +-- rw element-address
        |   +-- ...
        +-- M interfaces
```

The controller network model consists of the following key components:

- o A container with a list of topologies. A topology is a graph representation of a network at a particular layer, for example, an IS-IS topology, an overlay topology, or an Openflow topology. Specific topology types can be defined in their own separate YANG modules that augment the controller network model. Those augmentations are outside the scope of this example
- o An inventory of network elements, along with certain information that is mounted from each element. The information that is mounted in this case concerns interface configuration information. For this purpose, each list element that represents a network element contains a corresponding mountpoint. The mountpoint uses as its target the network element address information provided in the same list element
- o Each topology in turn contains a container with a list of nodes. A node is a network abstraction of a network device in the topology. A node is hosted on a network element, as indicated by a network-element leafref. This way, the "logical" and "physical" aspects of a node in the network are cleanly separated.
- o A node also contains a list of termination points that terminate links. A termination point is implemented on an interface. Therefore, it contains a leafref that references the corresponding interface configuration which is part of the mounted information of a network element. Again, the distinction between termination points and interfaces provides a clean separation between logical concepts at the network topology level and device-specific concepts that are instantiated at the level of a network element. Because the interface information is mounted from a different datastore and therefore occurs at a different level of the containment hierarchy than it would if it were not mounted, it is not possible to use the interface-ref type that is defined in YANG data model for interface management [] to allow the termination point refer to its supporting interface. For this reason, a new type definition "mountedIfRef" is introduced that allows to refer to interface information that is mounted and hence has a different path.
- o Finally, a topology also contains a container with a list of links. A link is a network abstraction that connects nodes via node termination points. In the example, directional point-to-point links are depicted in which one node termination point serves as source, another as destination.

The following is a YANG snippet of the module definition which makes use of the mountpoint definition.

```
<CODE BEGINS>
module controller-network {
  namespace "urn:cisco:params:xml:ns:yang:controller-network";
  // example only, replace with IANA namespace when assigned
  prefix cn;
  import mount {
    prefix mnt;
  }
  import interfaces {
    prefix if;
  }
  ...
  typedef mountedIfRef {
    type leafref {
      path "/cn:controller-network/cn:network-elements/"
        +"cn:network-element/cn:interfaces/if:interface/if:name";
      // cn:interfaces corresponds to the mountpoint
    }
  }
  ...
  list termination-point {
    key "tp-id";
    ...
    leaf ifref {
      type mountedIfRef;
    }
    ...
    list network-element {
      key "element-id";
      leaf element-id {
        type element-ID;
      }
      container element-address {
        ... // choice definition that allows to specify
        // host name,
        // IP addresses, URIs, etc
      }
      mnt:mountpoint "interfaces" {
        mnt:target "../element-address";
        mnt:subtree "/if:interfaces";
      }
      ...
    }
  }
  ...
}
<CODE ENDS>
```


Finally, the following contains an XML snippet of instantiated YANG information. We assume three datastores: NE1 and NE2 each have a datastore (the mount targets) that contains interface configuration data, which is mounted into NC's datastore (the mount client).

Interface information from NE1 datastore:

```
<interfaces>
  <interface>
    <name>fastethernet-1/0</name>
    <name>ethernetCsmacd</type>
    <location>1/0</location>
  </interface>
  <interface>
    <name>fastethernet-1/1</name>
    <name>ethernetCsmacd</type>
    <location>1/1</location>
  </interface>
</interfaces>
```

Interface information from NE2 datastore:

```
<interfaces>
  <interface>
    <name>fastethernet-1/0</name>
    <name>ethernetCsmacd</type>
    <location>1/0</location>
  </interface>
  <interface>
    <name>fastethernet-1/2</name>
    <name>ethernetCsmacd</type>
    <location>1/2</location>
  </interface>
</interfaces>
```

NC datastore with mounted interface information from NE1 and NE2:

```
<controller-network>
...
<network-elements>
  <network-element>
    <element-id>NE1</element-id>
    <element-address> .... </element-address>
    <interfaces>
      <if:interface>
        <if:name>fastethernet-1/0</if:name>
        <if:type>ethernetCsmacd</if:type>
        <if:location>1/0</if:location>
```

```
        </if:interface>
        <if:interface>
          <if:name>fastethernet-1/1</if:name>
          <if:type>ethernetCsmacd</if:type>
          <if:location>1/1</if:location>
        </if:interface>
      <interfaces>
    </network-element>
  <network-element>
    <element-id>NE2</element-id>
    <element-address> .... </element-address>
    <interfaces>
      <if:interface>
        <if:name>fastethernet-1/0</if:name>
        <if:type>ethernetCsmacd</if:type>
        <if:location>1/0</if:location>
      </if:interface>
      <if:interface>
        <if:name>fastethernet-1/2</if:name>
        <if:type>ethernetCsmacd</if:type>
        <if:location>1/2</if:location>
      </if:interface>
    <interfaces>
  </network-element>
</network-elements>
...
</controller-network>
```

Authors' Addresses

Alexander Clemm
Cisco Systems

E-Mail: alex@cisco.com

Jan Medved
Cisco Systems

E-Mail: jmedved@cisco.com

Eric Voit
Cisco Systems

E-Mail: evoit@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2014

A. Clemm
Cisco
H. Ananthakrishnan
Juniper Networks
J. Medved
T. Tkacik
Cisco
R. Varga
Pantheon Technologies SRO
N. Bahadur
Juniper Networks
October 21, 2013

A YANG Data Model for Network Topologies
draft-clemm-netmod-yang-network-topo-01.txt

Abstract

This document defines a YANG data model for network topologies.

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, 2014.

Copyright Notice

Copyright (c) 2013 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	2
2. Definitions and Acronyms	4
3. Network topology model overview	4
3.1. Model structure	5
3.2. Base model: Network Topology	5
3.2.1. Main building blocks	6
3.2.2. Discussion and selected design decisions	7
3.2.3. Open issues and items for further discussion	9
3.3. Extension of the model with specific topologies	9
3.3.1. Layer 3 Unicast - IGP	9
3.3.2. OSPF Topology	11
3.3.3. IS-IS Topology	14
3.3.4. TED - Traffic Engineering Data	16
4. Network Topology YANG module	16
5. Layer 3 Unicast IGP Topology YANG Module	23
6. OSPF Topology YANG Module	28
7. ISIS Topology YANG Module	32
8. TED YANG Module	35
9. Security Considerations	41
10. Contributors	42
11. Acknowledgements	42
12. References	42
12.1. Normative References	42
12.2. Informative References	42

1. Introduction

This document introduces a YANG [RFC6020] [RFC6021] data model for network topologies. The model allows an application to have a

holistic view of an entire network, all contained in a single conceptual YANG datastore.

In order to capture information that is specific to of a particular type of network topology, the basic model can be augmented and adapted. As a result, the data model is generic in nature and can be applied to many network topologies. For this reason, it is suitable for use as a general YANG data model framework to capture network topologies also beyond the types that are introduced here. Specific topology types that are covered in this document include Layer 3 Unicast IGP, IS-IS [RFC1195], and OSPF [RFC2178]. Adaptations and extensions to other types of topologies are possible, using similar model patterns to the ones that are illustrated.

There are multiple applications for such a data model. For example, a network controller can use the data model to represent the controller's view of a topology it controls and expose it to northbound applications via Netconf [RFC6241] or via a ReST Interface [I-D.bierman-netconf-restconf] [I-D.lhotka-netmod-yang-json]. Alternatively, nodes within the network can use the data model to capture their understanding of the overall network topology that they are contained in, as well as propagate this understanding and compare it with that of other nodes. The data model is generic in nature and can be applied to any type of network topology.

The data model is defined in several YANG modules:

- o Module "network-topology" contains a generic network topology model. It defines a network topology at its most general level of abstraction. It models aspects such as the nodes and edges that a topology graph is composed of, as well as termination points contained in the nodes that actually terminate the edges of the graph. A network can contain multiple topologies, for example topologies at different layers and overlay topologies. The model therefore allows also to capture the relationship between topologies, as well as the dependencies between nodes and termination points across topologies.
- o Module "l3-unicast-igp-topology" applies the general network topology model to Layer 3 Unicast IGP topologies. It augments the general topology with information specific to Layer 3 Unicast IGP. In doing so, it also illustrates the extension patterns associated with extending respectively augmenting the general topology model to meet the needs of a specific topology.

- o Module "ospf-topology" defines a topology model for OSPF, building on and extending the Layer 3 Unicast IGP topology model. It serves as an example of how the general topology model can be refined across multiple levels.
- o Module "isis-topology" defines a topology model for IS-IS, again building on and extending the Layer 3 Unicast IGP topology model.
- o Module "ted", finally, is a helper module, defining information kept in the Traffic Engineering Database (TED) that is leveraged by IS-IS and OSPF topologies.

2. Definitions and Acronyms

Datastore: A conceptual store of instantiated management information, with individual data items represented by data nodes which are arranged in hierarchical manner.

Data subtree: An instantiated data node and the data nodes that are hierarchically contained within it.

HTTP: Hyper-Text Transfer Protocol

IGP: Interior Gateway Protocol

IS-IS: Intermediate System to Intermediate System protocol

LSP: Label Switched Path

NETCONF: Network Configuration Protocol

OSPF: Open Shortest Path First, a link state routing protocol

URI: Uniform Resource Identifier

ReST: Representational State Transfer, a style of stateless interface and protocol that is generally carried over HTTP

SRLG: Shared Risk Link Group

TED: Traffic Engineering Database

YANG: A data definition language for NETCONF

3. Network topology model overview

This section provides an overview of the network topology model. We start with the structure of the foundational model that represents a

generic topology. Subsequently, an overview of the specific topologies is given - Layer 3 Unicast IGP, OSPF, and IS-IS, respectively. During the course of the discussion, selected design choices are explained and the pattern that should be applied to extend the model to new types of topologies is presented.

3.1. Model structure

The network topology model is defined by the following YANG modules, whose relationship is roughly depicted in the figure below.

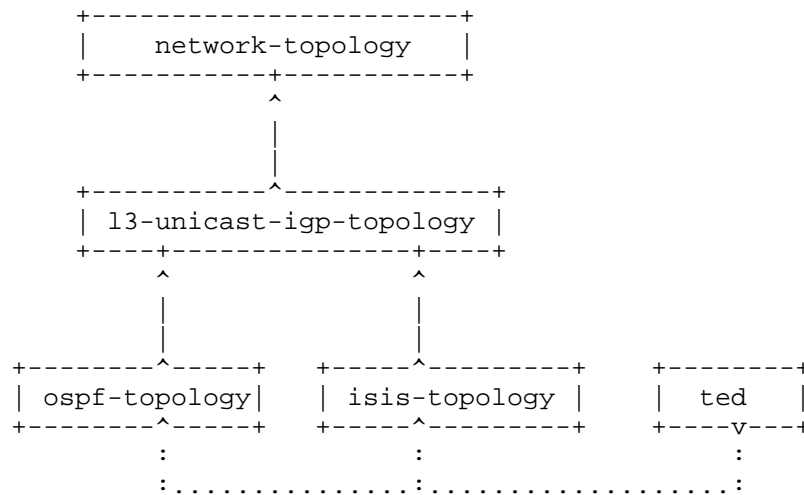


Figure 1: Overall model structure

YANG module `network-topology` defines the basic network topology model. YANG module `l3-unicast-igp-topology` builds on top of this model, augmenting `network-topology` with additional definitions needed to represent Layer 3 Unicast IGP topologies. This module in turn is augmented by YANG modules with additional definitions for OSPF and for IS-IS topologies, `ospf-topology` and `isis-topology`, respectively. Finally, YANG module `"ted"` contains a set of auxiliary definitions used by both `ospf-topology` and `isis-topology`, capturing data related to traffic engineering.

3.2. Base model: Network Topology

The structure of the network topology data model, as later defined in the YANG module "network-topology", is depicted in the following diagram. Brackets enclose list keys, "rw" means configuration data, "ro" means operational state data, and "?" designates optional nodes. The figure does not depict all definitions; it is intended to illustrate the overall structure.

```

module: network-topology
  +--rw network-topology
    +--rw topology [topology-id]
      +--rw topology-id          topology-id
      +--ro server-provided?      boolean
      +--rw topology-types
      +--rw underlay-topology [topology-ref]
        | +--rw topology-ref      topology-ref
      +--rw node [node-id]
        | +--rw node-id            node-id
        | +--rw supporting-node [node-ref]
        | | +--rw node-ref          node-ref
        | +--rw termination-point [tp-id]
        | | +--rw tp-id            tp-id
        | | +--ro tp-ref*          tp-ref
      +--rw link [link-id]
        +--rw link-id            link-id
        +--rw source
        | +--rw source-node        node-ref
        | +--rw source-tp?         tp-ref
        +--rw destination
        | +--rw dest-node          node-ref
        | +--rw dest-tp?           tp-ref
        +--rw supporting-link [link-ref]
          +--rw link-ref          link-ref

```

3.2.1. Main building blocks

A network can contain multiple topologies. Each topology is captured in its own list element, distinguished via a topology-id. This is captured by list "topology", contained underneath the root container for this module, "network-topology".

A topology has a certain type, such as OSPF or IS-IS. A topology can even have multiple types simultaneously. The type, or types, are captured underneath container "topology-types". This serves as container for data nodes that represent specific topology types. In this module, it serves merely as an augmentation target; topology-specific modules will later introduce new data nodes to represent new

topology types below this target, i.e. insert them below "topology-types" by ways of augmentation.

Topology types SHOULD always be represented using containers, not leafs of empty type. This allows to represent hierarchies of topology subtypes within the instance information. For example, an instance of an OSPF topology (which, at the same time, is a layer 3 unicast IGP topology) would contain underneath "topology-types" another container "l3-unicast-igp-topology", which in turn would contain a container "ospf-topology".

A topology can in turn be part of a hierarchy of topologies, building on top of other topologies. Any such topologies are captured in list "underlay-topology".

Furthermore, a topology contains nodes and links, each captured in their own list.

A node has a node-id. This distinguishes the node from other nodes in the list. In addition, a node has a list of termination points, used to terminate links. An examples of a termination point might be a physical or logical port or, more generally, an interface. Also, a node can in turn map onto other nodes in an underlay topology. This is captured in list "supporting-node".

A link is identified by a link-id, uniquely identifying the link within the topology. Links are point-to-point and unidirectional. Accordingly, a link contains a source and a destination. Both source and destination reference a corresponding node, as well as a termination point on that node. Analogous to a node, a link can in turn map onto other links an underlay topology. This is captured in list "supporting-link".

3.2.2. Discussion and selected design decisions

Rather than maintaining lists in separate containers, the model is kept relatively flat in terms of its containment structure. This way, path specifiers used to refer to specific nodes, be it in management operations or in specifications of constraints, can remain relatively compact. Of course, this means there is no separate structure in instance information that separates elements of different lists from one another. Such structure is semantically not required, although it might enhance human readability in some cases.

In an effort to minimize assumptions of what a topology might actually represent, mappings between topologies, nodes, links, and termination points are kept strictly generic. For example, no assumptions are made whether a termination point actually refers to

an interface, or whether a node refers to a specific "system" or device; the model at this generic level makes no provisions for that. Any greater specifics about mappings between upper and lower layers can be captured in augmenting modules. For example, if a termination point maps to an interface, an augmenting module can augment the termination point with a leaf that references the corresponding interface [I-D.ietf-netmod-interfaces-cfg]. If a node maps to a particular device or network element, an augmenting module can augment node with a leaf that references the network element.

The model makes extensive use of groupings, instead of simply defining data nodes "in-line". This allows to more easily include the corresponding data nodes in notifications, which then do not need to respecify each data node that is to be included. The tradeoff for this is that it makes the specification of constraints more complex, because constraints involving data nodes outside the grouping need to be specified in conjunction with a "uses" statement where the grouping is applied. This also means that constraints and XPath-statements need to be specified in such a way that they navigate "down" first and select entire sets of nodes, as opposed to being able to simply specify them against individual data nodes.

The topology model includes links that are point-to-point and unidirectional. It does not directly support multipoint and bidirectional links. While this may appear as a limitation, it does keep the model simple, generic, and allows it to very easily be subjected applications that make use of graph algorithms. Bidirectional connections can be represented through pairs of unidirectional links. By introducing hierarchies of nodes, with nodes at one level mapping onto a set of other nodes at another level, and the introducing new links for nodes at that level, topologies with connections representing non-point-to-point communication patterns can be represented.

Links are terminated by a single termination point, not sets of termination points. Connections involving multihoming or link aggregation schemes need to be represented using multiple point-to-point links, then defining a link at a higher layer that is supported by those individual links.

In a hierarchy of topologies, there are nodes mapping to nodes, links mapping to links, and termination points mapping to termination points. Some of this information is redundant. Specifically, with the link-to-links mapping known, and the termination points of each link known, maintaining separate termination point mapping information is not needed but can be derived via transitive closure. The model does provide for the option to include this information explicitly, but does not allow for it to be configured to avoid the

potential to introduce (and having to validate) corresponding integrity issues.

A topology's topology types are represented using a container which contains a data node for each of its topology types. A topology can encompass several types of topology simultaneously, hence a container is used instead of a case construct, with each topology type in turn represented by a dedicated presence container itself. The reason for not simply using an empty leaf, or even simpler, do away even with the topology container and just use a leaf-list of topology-type instead, is to be able to represent "class hierarchies" of topology types, with one topology type refining the other. Topology-type specific containers are to be defined in the topology-specific modules, augmenting the topology-types container.

3.2.3. Open issues and items for further discussion

YANG requires data needs to be designated as either configuration or operational data, but not both, yet it is important to have all topology information, including vertical cross-topology dependencies, captured in one coherent model. In most cases topology information is discovered about a network; the topology is considered a property of the network that is reflected in the model. That said, it is conceivable that certain types of topology need to also be configurable by an application.

There are several alternatives in which this can be addressed. The alternative chosen in this draft does not restrict topology information as read-only, but includes a flag that indicates for each topology whether it should be considered as read-only or configurable by applications.

An alternative would be to designate topology list elements as read only. The read-only topology list includes each topology; it is the complete reference. In parallel a second topology list is introduced. This list serves the purpose of being able to configure topologies which are then mirrored in the read-only list. The configurable topology list adheres to the same structure and uses the same groupings as its read-only counterpart. As most data is defined in those groupings, the amount of additional definitions required will be limited. A configurable topology will thus be represented twice: once in the read-only list of all topologies, a second time in a configuration sandbox.

3.3. Extension of the model with specific topologies

3.3.1. Layer 3 Unicast - IGP

In order to represent a general Layer 3 Unicast IGP topology, the basic network topology model needs to be extended. The corresponding extensions are introduced in a separate YANG module "l3-unicast-igp-topology". The structure of those extensions is depicted in the following diagram. Brackets enclose list keys, "rw" means configuration, "ro" operational state data, "?" designates optional nodes, "*" designates nodes that can have multiple instances. Parantheses enclose choice and case nodes. Data nodes from the network-topology module are omitted (indicated by "....."), as long as not required to indicate containment structure. Notifications are not depicted.

```

module: network-topology
  +--rw network-topology
    +--rw topology [topology-id]
      +.....
      +--rw topology-types
        |   +--rw l3t:l3-unicast-igp-topology?
        |   .....
        +--rw node [node-id]
          |   .....
          |   +--rw termination-point [tp-id]
          |     |   .....
          |     |   +--rw l3t:igp-termination-point-attributes
          |     |     +--rw (termination-point-type)?
          |     |       +--:(ip)
          |     |         |   +--rw l3t:ip-address*      inet:ip-address
          |     |         +--:(unnumbered)
          |     |           +--rw l3t:unnumbered-id?    uint32
          |     +--rw l3t:igp-node-attributes
          |       +--rw l3t:name?      inet:domain-name
          |       +--rw l3t:flag*     flag-type
          |       +--rw l3t:router-id* inet:ip-address
          |       +--rw l3t:prefix [prefix]
          |         +--rw l3t:prefix  inet:ip-prefix
          |         +--rw l3t:metric? uint32
          |         +--rw l3t:flag*   flag-type
          +--rw link [link-id]
            |   .....
            |   +--rw l3t:igp-link-attributes
            |     +--rw l3t:name?      string
            |     +--rw l3t:flag*     flag-type
            |     +--rw l3t:metric?   uint32
            +--rw l3t:igp-topology-attributes
              +--rw l3t:name?      string
              +--rw l3t:flag*     flag-type

```

The module augments the original network-topology module as follows:

- o A new topology type is introduced, l3-unicast-igp-topology-type. This is represented by a container object, which is inserted under the "topology-types" container of the network topology module.
- o Additional topology attributes are introduced, defined in a grouping, which augments the "topology" list of the network topology module. The attributes include an IGP name, as well as a set of flags (represented through a leaf-list). Each type of flag is represented by a separate identity. This allows to introduce additional flags in augmenting modules that are associated with specific IGP topologies, without needing to revise this module.
- o Additional data objects for nodes are introduced by augmenting the "node" list of the network topology module. New objects include again a set of flags, as well as a list of prefixes. Each prefix in turn includes an ip prefix, a metric, and a prefix-specific set of flags.
- o Links are augmented as well with a set of parameters, allowing to associate a link with an IGP name, another set of flags, and a link metric.

In addition, the module defines a set of notifications to alert clients of any events concerning links, nodes, prefixes, and termination points. Each notification includes an indication of the type of event, the topology from which it originated, and the affected node, or link, or prefix, or termination point. In addition, as a convenience to applications, additional data of the affected node, or link, or termination point (respectively) is included. While this makes notifications larger in volume than they would need to be, it avoids the need for subsequent retrieval of context information, which also might have changed in the meantime.

3.3.2. OSPF Topology

OSPF is the next type of topology represented in the model. OSPF represents a particular type of Layer 3 Unicast IGP. Accordingly, this time the Layer 3 Unicast IGP topology model needs to be extended. The corresponding extensions are introduced in a separate YANG module "ospf-topology", whose structure is depicted in the following diagram. For the most part, this module augments "l3-unicast-igp-topology". Like before, brackets enclose list keys, "rw" means configuration, "ro" operational state data, "?" designates optional nodes, "*" designates nodes that can have multiple instances. Parentheses enclose choice and case nodes. Data nodes from the network-topology module are omitted (indicated by "....."),

as long as not required to indicate containment structure.
Notifications are not depicted.

```

module: network-topology
  +--rw network-topology
    +--rw topology [topology-id]
      .....
      +--rw topology-types
        |   +--rw l3t:l3-unicast-igp-topology?
        |   +--rw ospf:ospf?
        |   .....
      +--rw node [node-id]
        |   .....
        |   +--rw l3t:igp-node-attributes
        |     |   .....
        |     |   +--rw l3t:prefix [prefix]
        |     |     |   +.....
        |     |     |   +--rw ospf:ospf-prefix-attributes
        |     |     |     +--rw ospf:forwarding-address?   inet:ipv4-address
        |     |   +--rw ospf:ospf-node-attributes
        |     |     +--rw (router-type)?
        |     |       |   +--:(abr)
        |     |       |   |   +--rw ospf:abr?               empty
        |     |       |   +--:(asbr)
        |     |       |   |   +--rw ospf:asbr?              empty
        |     |       |   +--:(internal)
        |     |       |   |   +--rw ospf:internal?          empty
        |     |       |   +--:(pseudonode)
        |     |       |   |   +--rw ospf:pseudonode?         empty
        |     |     +--rw ospf:dr-interface-id?             uint32
        |     |     +--rw ospf:multi-topology-id*            uint8
        |     |     +--rw ospf:capabilities?                 bits
        |     |     +--rw ospf:ted
        |     |       +--rw ospf:te-router-id-ipv4?          inet:ipv4-address
        |     |       +--rw ospf:te-router-id-ipv6?          inet:ipv6-address
        |     |       +--rw ospf:ipv4-local-address [ipv4-prefix]
        |     |       |   +--rw ospf:ipv4-prefix             inet:ipv4-prefix
        |     |       +--rw ospf:ipv6-local-address [ipv6-prefix]
        |     |       |   +--rw ospf:ipv6-prefix             inet:ipv6-prefix
        |     |       |   +--rw ospf:prefix-option?          uint8
        |     |     +--rw ospf:pcc-capabilities?             pcc-capabilities
        |     |     .....
      +--rw link [link-id]
        |   .....
        |   +--rw l3t:igp-link-attributes
        |     |   .....
        |     |   +--rw ospf:ospf-link-attributes
        |     |     +--rw ospf:multi-topology-id?            uint8

```

```

|
|
|      +---rw ospf:ted
|      +---rw ospf:color?                uint32
|      +---rw ospf:max-link-bandwidth?    decimal64
|      +---rw ospf:max-resv-link-bandwidth? decimal64
|      +---rw ospf:unreserved-bandwidth [priority]
|      |   +---rw ospf:priority          uint8
|      |   +---rw ospf:bandwidth?        decimal64
|      +---rw ospf:te-default-metric?      uint32
|      +---rw ospf:srlg
|      +---rw ospf:interface-switching-capabilities [switching
-capability]
|      |   +---rw ospf:switching-capability          ted:s
witching-capabilities
|      |   +---rw ospf:encoding?                    uint8
|      |   +---rw ospf:max-lsp-bandwidth [priority]
|      |   |   +---rw ospf:priority          uint8
|      |   |   +---rw ospf:bandwidth?        decimal64
|      |   +---rw ospf:packet-switch-capable
|      |   |   +---rw ospf:minimum-lsp-bandwidth? decimal64
|      |   |   +---rw ospf:interface-mtu?        uint16
|      |   +---rw ospf:time-division-multiplex-capable
|      |   |   +---rw ospf:minimum-lsp-bandwidth? decimal64
|      |   |   +---rw ospf:indication?          uint16
|      |   +---rw ospf:srlg-values [srlg-value]
|      |   |   +---rw ospf:srlg-value          uint32
|      |   +---rw ospf:link-protection-type?      uint16
|      .....
+---rw l3t:igp-topology-attributes
      .....
      +---rw ospf:ospf-topology-attributes
      |   +---rw ospf:area-id?    area-id
      .....

```

The module augments "l3-unicast-igp-topology" as follows:

- o A new topology type for an OSPF topology is introduced. This is represented by a container object, which is inserted under the "l3-unicast-igp-topology" container of the l3-unicast-igp-topology module. This way, an ospf topology represents both a l3-unicast-igp topology and an ospf topology.
- o Additional topology attributes are defined in a new grouping which augments igp-topology-attributes of the l3-unicast-igp-topology module. The attributes include an OSPF area-id identifying the OSPF area.
- o Additional data objects for nodes are introduced by augmenting the igp-node-attributes of the l3-unicast-igp-topology module. New objects include router-type, de-interface-id for pseudonodes, list

of multi-topology-ids, ospf node capabilities and traffic engineering attributes.

- o Links are augmented with a multi-topology-id and traffic engineering link attributes.
- o Prefixes are augmented with OSPF specific forwarding address.

In addition, the module extends IGP node, link and prefix notifications with OSPF attributes.

3.3.3. IS-IS Topology

IS-IS is another type of Layer 3 Unicast IGP. Like OSPF topology, IS-IS topology is defined in a separate module, "isis-topology", which augments "l3-unicast-igp-topology". The structure is depicted in the following diagram. Like before, brackets enclose list keys, "rw" means configuration, "ro" operational state data, "?" designates optional nodes, "*" designates nodes that can have multiple instances. Parentheses enclose choice and case nodes. Data nodes from the network-topology module are omitted (indicated by "....."), as long as not required to indicate containment structure. Notifications are not depicted.

```

module: network-topology
  +--rw network-topology
    +--rw topology [topology-id]
      .....
      |   +--rw l3t:l3-unicast-igp-topology?
      |   .....
      |   |   +--rw isis:isis?
      |   |   .....
      |   +--rw node [node-id]
      |   .....
      |   |   +--rw l3t:igp-node-attributes
      |   |   .....
      |   |   +--rw isis:isis-node-attributes
      |   |   |   +--rw isis:iso
      |   |   |   |   +--rw isis:iso-system-id?       iso-system-id
      |   |   |   |   +--rw isis:iso-pseudonode-id?   iso-pseudonode-id
      |   |   |   +--rw isis:net*                     iso-net-id
      |   |   |   +--rw isis:multi-topology-id*       uint8
      |   |   |   +--rw (router-type)?
      |   |   |   |   +--:(level-2)
      |   |   |   |   |   +--rw isis:level-2?         empty
      |   |   |   |   +--:(level-1)
      |   |   |   |   |   +--rw isis:level-1?         empty
      |   |   |   |   +--:(level-1-2)

```



```

|         +--rw isis:level-1-2?          empty
+--rw isis:ted
|   +--rw isis:te-router-id-ipv4?      inet:ipv4-address
|   +--rw isis:te-router-id-ipv6?      inet:ipv6-address
|   +--rw isis:ipv4-local-address [ipv4-prefix]
|   |   +--rw isis:ipv4-prefix      inet:ipv4-prefix
|   +--rw isis:ipv6-local-address [ipv6-prefix]
|   |   +--rw isis:ipv6-prefix      inet:ipv6-prefix
|   |   +--rw isis:prefix-option?    uint8
|   +--rw isis:pcc-capabilities?      pcc-capabilities
+--rw link [link-id]
|   .....
|   +--rw l3t:igp-link-attributes
|   |   .....
|   |   +--rw isis:isis-link-attributes
|   |   |   +--rw isis:multi-topology-id?    uint8
|   |   |   +--rw isis:ted
|   |   |   |   +--rw isis:color?            uint32
|   |   |   |   +--rw isis:max-link-bandwidth?    decimal64
|   |   |   |   +--rw isis:max-resv-link-bandwidth?    decimal64
|   |   |   |   +--rw isis:unreserved-bandwidth [priority]
|   |   |   |   |   +--rw isis:priority      uint8
|   |   |   |   |   +--rw isis:bandwidth?    decimal64
|   |   |   |   +--rw isis:te-default-metric?    uint32
|   |   |   +--rw isis:srlg
|   |   |   |   +--rw isis:interface-switching-capabilities [switching
-capability]
witching-capabilities |   +--rw isis:switching-capability          ted:s
|   |   +--rw isis:encoding?          uint8
|   |   +--rw isis:max-lsp-bandwidth [priority]
|   |   |   +--rw isis:priority      uint8
|   |   |   +--rw isis:bandwidth?    decimal64
|   |   +--rw isis:packet-switch-capable
|   |   |   +--rw isis:minimum-lsp-bandwidth?    decimal64
|   |   |   +--rw isis:interface-mtu?          uint16
|   |   +--rw isis:time-division-multiplex-capable
|   |   |   +--rw isis:minimum-lsp-bandwidth?    decimal64
|   |   |   +--rw isis:indication?          uint16
|   |   +--rw isis:srlg-values [srlg-value]
|   |   |   +--rw isis:srlg-value      uint32
|   |   +--rw isis:link-protection-type?          uint16
+--rw l3t:igp-topology-attributes
|   .....
|   +--rw isis:isis-topogloy-attributes
|   |   +--rw isis:net?    iso-net-id

```

The module augments the l3-unicast-igp-topology as follows:

- o A new topology type is introduced, "isis-topology-type". This is represented by a container object, which is inserted under the "l3-unicast-igp-topology" container of the l3-unicast-igp-topology module. This way, an isis topology represents both a l3-unicast-igp-topology and an isis topology.
- o Additional topology attributes are introduced in a new grouping which augments "igp-topology-attributes" of the l3-unicast-igp-topology module. The attributes include an ISIS NET-id identifying the area.
- o Additional data objects for nodes are introduced by augmenting "igp-node-attributes" of the l3-unicast-igp-topology module. New objects include router-type, iso-system-id to identify the router, a list of multi-topology-id, a list of NET ids, and traffic engineering attributes.
- o Links are augmented with multi-topology-id and traffic engineering link attributes.

In addition, the module augments IGP nodes and links with ISIS attributes.

3.3.4. TED - Traffic Engineering Data

Traffic Engineering Data is required both by OSPF and IS-IS, which are defined in separate modules. Information shared by both is defined in another module, "ted". This module defines a set of groupings with auxiliary information required and shared by those other modules. This module details traffic-engineering node and link attributes:

- o TED node attributes include te-router-id for IPv4 and IPv6, local IPv4 and IPv6 addresses and path computation client capabilities. The path computation client capabilities in turn include a bit vector for various path computation capabilities.
- o TED link attributes comprise link color, max-link-bandwidth, max-resv-link-bandwidth, unreserved bandwidth and re-metric. They also include SRLG attributes which contains interface switching capabilities, a list of SRLG values, and a link protection type. The interface switching capabilities in turn contain a list element for each switching capability, defining encoding, max-lsp-bandwidth, and interface switching specific attributes.

4. Network Topology YANG module

<CODE BEGINS>

```
file "network-topology@2013-10-21.yang"
module network-topology {
  yang-version 1;
  namespace "urn:TBD:params:xml:ns:yang:network-topology";
  // replace with IANA namespace when assigned
  prefix "nt";

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

  organization "TBD";

  contact "WILL-BE-DEFINED-LATER";

  description
    "This module defines a model for the topology of a network.
    Key design decisions are as follows:
    A topology consists of a set of nodes and links.
    Links are point-to-point and unidirectional.
    Bidirectional connections need to be represented through
    two separate links.
    Multipoint connections, broadcast domains etc can be represented
    through a hierarchy of nodes, then connecting nodes at
    upper layers of the hierarchy.";

  revision 2013-10-21 {
    description
      "Initial revision.";
  }

  typedef topology-id {
    type inet:uri;
    description
      "An identifier for a topology.";
  }

  typedef node-id {
    type inet:uri;
    description
      "An identifier for a node in a topology.
      The identifier may be opaque.
      The identifier SHOULD be chosen such that the same node in a
      real network topology will always be identified through the
      same identifier, even if the model is instantiated in separate
      datastores. An implementation MAY choose to capture semantics
      in the identifier, for example to indicate the type of node
      and/or the type of topology that the node is a part of.";
  }
}
```

```
typedef link-id {
    type inet:uri;
    description
        "An identifier for a link in a topology.
        The identifier may be opaque.
        The identifier SHOULD be chosen such that the same link in a
        real network topology will always be identified through the
        same identifier, even if the model is instantiated in separate
        datastores. An implementation MAY choose to capture semantics
        in the identifier, for example to indicate the type of link
        and/or the type of topology that the link is a part of.";
}

typedef tp-id {
    type inet:uri;
    description
        "An identifier for termination points on a node.
        The identifier may be opaque.
        The identifier SHOULD be chosen such that the same TP in a
        real network topology will always be identified through the
        same identifier, even if the model is instantiated in separate
        datastores. An implementation MAY choose to capture semantics
        in the identifier, for example to indicate the type of TP
        and/or the type of node and topology that the TP is a part of.";
}

typedef tp-ref {
    type leafref {
        path "/network-topology/topology/node/termination-point/tp-id";
    }
    description
        "A type for an absolute reference to a termination point.
        (This type should not be used for relative references.
        In such a case, a relative path should be used instead.);";
}

typedef topology-ref {
    type leafref {
        path "/network-topology/topology/topology-id";
    }
    description
        "A type for an absolute reference a topology instance.";
}

typedef node-ref {
    type leafref {
        path "/network-topology/topology/node/node-id";
    }
    description
```

```
        "A type for an absolute reference to a node instance.
        (This type should not be used for relative references.
        In such a case, a relative path should be used instead.);";
    }

typedef link-ref {
    type leafref {
        path "/network-topology/topology/link/link-id";
    }
    description
        "A type for an absolute reference a link instance.
        (This type should not be used for relative references.
        In such a case, a relative path should be used instead.);";
}

grouping tp-attributes {
    description
        "The data objects needed to define a termination point.
        (This only includes a single leaf at this point, used
        to identify the termination point.)
        Provided in a grouping so that in addition to the datastore,
        the data can also be included in notifications.";
    leaf tp-id {
        type tp-id;
    }
    leaf-list tp-ref {
        type tp-ref;
        config false;
        description
            "The leaf list identifies any termination points that the
            termination point is dependent on, or maps onto.
            Those termination points will themselves be contained
            in a supporting node.
            This dependency information can be inferred from
            the dependencies between links. For this reason,
            this item is not separately configurable. Hence no
            corresponding constraint needs to be articulated.
            The corresponding information is simply provided by the
            implementing system.";
    }
}

grouping node-attributes {
    description
        "The data objects needed to define a node.
        The objects are provided in a grouping so that in addition to
        the datastore, the data can also be included in notifications
        as needed.";
```

```

    leaf node-id {
        type node-id;
        description
            "The identifier of a node in the topology.
            A node is specific to a topology to which it belongs.";
    }
    list supporting-node {
        description
            "This list defines vertical layering information for nodes.
            It allows to capture for any given node, which node (or nodes)
            in the corresponding underlay topology it maps onto.
            A node can map to zero, one, or more nodes below it;
            accordingly there can be zero, one, or more elements in the li
st.

            If there are specific layering requirements, for example
            specific to a particular type of topology that only allows
            for certain layering relationships, the choice
            below can be augmented with additional cases.
            A list has been chosen rather than a leaf-list in order
            to provide room for augmentations, e.g. for
            statistics or prioritization information associated with
            supporting nodes.";
        key "node-ref";
        leaf node-ref {
            type node-ref;
        }
    }
}

grouping link-attributes {
    // This is a grouping, not defined inline with the link definition its
elf,
    // so it can be included in a notification, if needed
    leaf link-id {
        type link-id;
        description
            "The identifier of a link in the topology.
            A link is specific to a topology to which it belongs.";
    }
    container source {
        leaf source-node {
            mandatory true;
            type node-ref;
            description
                "Source node identifier, must be in same topology.";
        }
        leaf source-tp {
            type tp-ref;
            description
                "Termination point within source node that terminates the
link.";

```

```

    }
  }
  container destination {
    leaf dest-node {
      mandatory true;
      type node-ref;
      description
        "Destination node identifier, must be in same topology.";
    }
    leaf dest-tp {
      type tp-ref;
      description
        "Termination point within destination node that terminates
the link.";
    }
  }
  list supporting-link {
    key "link-ref";
    leaf link-ref {
      type link-ref;
    }
  }
}

container network-topology {
  list topology {
    description "
      This is the model of an abstract topology.
      A topology contains nodes and links.
      Each topology MUST be identified by
      unique topology-id for reason that a network could contain man
y
      topologies.
    ";
    key "topology-id";
    leaf topology-id {
      type topology-id;
      description "
        It is presumed that a datastore will contain many topologi
es. To
        distinguish between topologies it is vital to have UNIQUE
        topology identifiers.
      ";
    }
    leaf server-provided {
      type boolean;
      config false;
      description "
        Indicates whether the topology is configurable by clients,
        or whether it is provided by the server. This leaf is

```

```

        populated by the server implementing the model.
        It is set to false for topologies that are created by a client;
        it is set to true otherwise. If it is set to true, any
        attempt to edit the topology MUST be rejected.
    ";
}
container topology-types {
    description
        "This container is used to identify the type, or types
        (as a topology can support several types simultaneously),
        of the topology.
        Topology types are the subject of several integrity constraints
        that an implementing server can validate in order to
        maintain integrity of the datastore.
        Topology types are indicated through separate data nodes;
        the set of topology types is expected to increase over time.

        To add support for a new topology, an augmenting module
        needs to augment this container with a new empty optional
        container to indicate the new topology type.
        The use of a container allows to indicate a subcategorization
        of topology types.
        The container SHALL NOT be augmented with any data nodes
        that serve a purpose other than identifying a particular
        topology type.
    ";
}
list underlay-topology {
    key "topology-ref";
    leaf topology-ref {
        type topology-ref;
    }
    // a list, not a leaf-list, to allow for potential augmentation
    // with properties specific to the underlay topology,
    // such as statistics, preferences, or cost.
    description
        "Identifies the topology, or topologies, that this topology
        is dependent on.";
}

list node {
    description "The list of network nodes defined for the topology.";
    key "node-id";
    uses node-attributes;
    must "boolean(..../underlay-topology[*]/node[../supporting-nodes/
node-ref]))";
    // This constraint is meant to ensure that a referenced node
    // is in fact
    // a node in an underlay topology.
    list termination-point {
        description
```



```

        "A termination point can terminate a link.
        Depending on the type of topology, a termination point
could,
        for example, refer to a port or an interface.";
        key "tp-id";
        uses tp-attributes;
    }
}

list link {
    description "
        A Network Link connects a by Local (Source) node and
        a Remote (Destination) Network Nodes via a set of the
        nodes' termination points.
        As it is possible to have several links between the same
        source and destination nodes, and as a link could potentia
lly
        be re-homed between termination points, to ensure that we
        would always know to distinguish between links, every link
        is identified by a dedicated link identifier.
        Note that a link models a point-to-point link, not a multi
point
        link.
        Layering dependencies on links in underlay topologies are
        not represented as the layering information of nodes and o
f
        termination points is sufficient.
    ";
    key "link-id";
    uses link-attributes;
    must "boolean(..underlay-topology/link[../supporting-link]);
        // Constraint: any supporting link must be part of an unde
rly topology
    must "boolean(..node[../source/source-node]);
        // Constraint: A link must have as source a node of the sa
me topology
    must "boolean(..node[../destination/dest-node]);
        // Constraint: A link must have as source a destination of
the same topology
    must "boolean(..node/termination-point[../source/source-tp]);
        // Constraint: The source termination point must be contai
ned in the source node
    must "boolean(..node/termination-point[../destination/dest-tp]
)";
        // Constraint: The destination termination point must be c
ontained
        // in the destination node
    }
}
}
}

<CODE ENDS>
```

5. Layer 3 Unicast IGP Topology YANG Module

<CODE BEGINS>


```
file "l3-unicast-igp-topology@2013-10-21.yang"
module l3-unicast-igp-topology {
    yang-version 1;
    namespace "urn:TBD:params:xml:ns:yang:nt:l3-unicast-igp-topology";
    // replace with IANA namespace when assigned
    prefix "l3t";
    import network-topology {
        prefix "nt";
    }

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

    organization "TBD";
    contact "TBD";

    revision "2013-10-21" {
        description "Initial revision";
        reference "TBD";
    }

    typedef igp-event-type {
        description "IGP Event type for notifications";
        type enumeration {
            enum "add" {
                value 0;
                description "An IGP node or link or prefix or terminat
ion-point has been added";
            }
            enum "remove" {
                value 1;
                description "An IGP node or link or prefix or termination-
point has been removed";
            }
            enum "update" {
                value 2;
                description "An IGP node or link or prefix or termination-
point has been updated";
            }
        }
    } // igp-event-type

    identity flag-identity {
        description "Base type for flags";
    }
    identity undefined-flag {
        base "flag-identity";
    }

    typedef flag-type {
```

```
        type identityref {
            base "flag-identity";
        }
    }

    grouping igp-prefix-attributes {
        leaf prefix {
            type inet:ip-prefix;
        }
        leaf metric {
            type uint32;
        }
        leaf-list flag {
            type flag-type;
        }
    }

    grouping l3-unicast-igp-topology-type {
        container l3-unicast-igp-topology {
            presence "indicates L3 Unicast IGP Topology";
        }
    }

    grouping igp-topology-attributes {
        container igp-topology-attributes {
            leaf name {
                description "Name of the topology";
                type string;
            }
            leaf-list flag {
                description "Topology flags";
                type flag-type;
            }
        }
    }

    grouping igp-node-attributes {
        container igp-node-attributes {
            leaf name {
                description "Node name";
                type inet:domain-name;
            }
            leaf-list flag {
                description "Node operational flags";
                type flag-type;
            }
            leaf-list router-id {
                description "Router-id for the node";
            }
        }
    }
```

```

        type inet:ip-address;
    }
    list prefix {
        key "prefix";
        uses igp-prefix-attributes;
    }
}

grouping igp-link-attributes {
    container igp-link-attributes {
        leaf name {
            description "Link Name";
            type string;
        }
        leaf-list flag {
            description "Link flags";
            type flag-type;
        }
        leaf metric {
            description "Link Metric";
            type uint32 {
                range "0..16777215" {
                    description "
                    ";
                    // OSPF/ISIS supports max 3 byte metric.
                    // Ideally we would like this restriction to be
                    // defined in the derived models, however,
                    // we are not allowed to augment a "must" statement.
                }
            }
        }
    }
}

} // grouping igp-link-attributes

grouping igp-termination-point-attributes {
    container igp-termination-point-attributes {
        choice termination-point-type {
            case ip {
                leaf-list ip-address {
                    description "IPv4 or IPv6 address";
                    type inet:ip-address;
                }
            }
            case unnumbered {
                leaf unnumbered-id {
                    description "Unnumbered interface identifier";
                    type uint32;
                }
            }
        }
    }
}

```

```

        }
    }
} // grouping igp-termination-point-attributes

augment "/nt:network-topology/nt:topology/nt:topology-types" {
    uses l3-unicast-igp-topology-type;
}

augment "/nt:network-topology/nt:topology" {
    when "nt:topology-types/l3-unicast-igp-topology";
    uses igp-topology-attributes;
}

augment "/nt:network-topology/nt:topology/nt:node" {
    when "../nt:topology-types/l3-unicast-igp-topology";
    uses igp-node-attributes;
}

augment "/nt:network-topology/nt:topology/nt:link" {
    when "../nt:topology-types/l3-unicast-igp-topology";
    uses igp-link-attributes;
}

augment "/nt:network-topology/nt:topology/nt:node/nt:termination-point" {
    when "../../../nt:topology-types/l3-unicast-igp-topology";
    uses igp-termination-point-attributes;
}

notification igp-node-event {
    leaf igp-event-type {
        type igp-event-type;
    }
    leaf topology-ref {
        type nt:topology-ref;
    }
    uses l3-unicast-igp-topology-type;
    uses nt:node-attributes;
    uses igp-node-attributes;
}

notification igp-link-event {
    leaf igp-event-type {
        type igp-event-type;
    }
    leaf topology-ref {
        type nt:topology-ref;
    }
}

```

```
        uses l3-unicast-igp-topology-type;
        uses nt:link-attributes;
        uses igp-link-attributes;
    }

    notification igp-prefix-event {
        leaf igp-event-type {
            type igp-event-type;
        }
        leaf topology-ref {
            type nt:topology-ref;
        }
        leaf node-ref {
            type nt:node-ref;
        }
        uses l3-unicast-igp-topology-type;
        container prefix {
            uses igp-prefix-attributes;
        }
    }

    notification termination-point-event {
        leaf igp-event-type {
            type igp-event-type;
        }
        leaf topology-ref {
            type nt:topology-ref;
        }
        leaf node-ref {
            type nt:node-ref;
        }
        uses l3-unicast-igp-topology-type;
        uses nt:tp-attributes;
        uses igp-termination-point-attributes;
    }
}

<CODE ENDS>
```

6. OSPF Topology YANG Module

```
<CODE BEGINS>
file "ospf-topology@2013-10-21.yang"
module ospf-topology {
    yang-version 1;
    namespace "urn:TBD:params:xml:ns:yang:ospf-topology";
    // replace with IANA namespace when assigned
```



```
    prefix "ospf";

    import network-topology {
        prefix "nt";
    }

    import l3-unicast-igp-topology {
        prefix "l3t";
    }
    import ietf-inet-types {
        prefix "inet";
    }
    import ted {
        prefix "ted";
    }

    organization "TBD";
    contact "TBD";
    description "OSPF Topology model";

    revision "2013-10-21" {
        description "Initial revision";
        reference "TBD";
    }

    typedef area-id {
        description "OSPF Area ID";
        type uint32;
    }

    grouping ospf-topology-type {
        container ospf {
            presence "indiates OSPF Topology";
        }
    }

    augment "/nt:network-topology/nt:topology/nt:topology-types/l3t:l3-unicast-igp-topology" {
        uses ospf-topology-type;
    }

    augment "/nt:network-topology/nt:topology/l3t:igp-topology-attributes" {
        when "../nt:topology-types/l3t:l3-unicast-igp-topology/ospf";
        container ospf-topology-attributes {
            leaf area-id {
                type area-id;
            }
        }
    }
}
```

```

    augment "/nt:network-topology/nt:topology/nt:node/l3t:igp-node-attributes"
    {
        when "../../../nt:topology-types/l3t:l3-unicast-igp-topology/ospf";
        uses ospf-node-attributes;
    }

    augment "/nt:network-topology/nt:topology/nt:link/l3t:igp-link-attributes"
    {
        when "../../../nt:topology-types/l3t:l3-unicast-igp-topology/ospf";
        uses ospf-link-attributes;
    }

    augment "/nt:network-topology/nt:topology/nt:node/l3t:igp-node-attributes/
l3t:prefix" {
        when "../../../nt:topology-types/l3t:l3-unicast-igp-topology/ospf";
        uses ospf-prefix-attributes;
    }

    grouping ospf-node-attributes {
        container ospf-node-attributes {
            choice router-type {
                case abr {
                    leaf abr {
                        type empty;
                    }
                }
                case asbr {
                    leaf asbr {
                        type empty;
                    }
                }
                case internal {
                    leaf internal {
                        type empty;
                    }
                }
                case pseudonode {
                    leaf pseudonode {
                        type empty;
                    }
                }
            }
        }
        leaf dr-interface-id {
            when "../../../router-type/pseudonode";
            description "For pseudonodes, DR interface-id";
            default "0";
            type uint32;
        }
        leaf-list multi-topology-id {
            description "List of Multi-Topology Identifier up-to 128 (0-12
7). RFC 4915";
            max-elements "128";

```

```
        type uint8 {
            range "0..127";
        }
    }
    leaf capabilities {
        description "OSPF capabilities as bit vector. RFC 4970";
        type bits {
            bit graceful-restart-capable {
                position 0;
            }
            bit graceful-restart-helper {
                position 1;
            }
            bit stub-router-support {
                position 2;
            }
            bit traffic-engineering-support {
                position 3;
            }
            bit point-to-point-over-lan {
                position 4;
            }
            bit experimental-te {
                position 5;
            }
        }
    }
    container ted {
        uses ted:ted-node-attributes;
    }
} // ospf
} // ospf-node-attributes

grouping ospf-link-attributes {
    container ospf-link-attributes {
        leaf multi-topology-id {
            type uint8 {
                range "0..127";
            }
        }
        container ted {
            uses ted:ted-link-attributes;
        }
    }
} // ospf-link-attributes

grouping ospf-prefix-attributes {
    container ospf-prefix-attributes {
```

```
        leaf forwarding-address {
            when ".../l3t:l3-unicast-igp-topology/l3t:ospf/l3t:router-ty
pe/l3t:asbr";
            type inet:ipv4-address;
        }
    }

    augment "/l3t:igp-node-event" {
        uses ospf-topology-type;
        uses ospf:ospf-node-attributes;
    }

    augment "/l3t:igp-link-event" {
        uses ospf-topology-type;
        uses ospf:ospf-link-attributes;
    }

    augment "/l3t:igp-prefix-event" {
        uses ospf-topology-type;
        uses ospf:ospf-prefix-attributes;
    }
}

<CODE ENDS>
```

7. ISIS Topology YANG Module

```
<CODE BEGINS>
file "isis-topology@2013-10-21.yang"
module isis-topology {
    yang-version 1;
    namespace "urn:TBD:params:xml:ns:yang:network:isis-topology";
    // replace with IANA namespace when assigned
    prefix "isis";
    import network-topology {
        prefix nt;
    }
    import l3-unicast-igp-topology {
        prefix igp;
    }
    import ted {
        prefix ted;
    }

    organization "TBD";
    contact "TBD";
    description "ISIS Topology model";
```

```

revision "2013-10-21" {
    description "Initial version";
}
typedef iso-system-id {
    description "ISO System ID. RFC 1237";
    type string {
        pattern '[0-9a-fA-F]{4}(\.[0-9a-fA-F]{4}){2}';
    }
}

typedef iso-pseudonode-id {
    description "ISO pseudonode id for broadcast network";
    type string {
        pattern '[0-9a-fA-F]{2}';
    }
}

typedef iso-net-id {
    description "ISO NET ID. RFC 1237";
    type string {
        pattern '[0-9a-fA-F]{2}((\.[0-9a-fA-F]{4}){6})';
    }
}

grouping isis-topology-type {
    container isis {
        presence "Indicates ISIS Topology";
    }
}

augment "/nt:network-topology/nt:topology/nt:topology-types/igp:l3-unicast-igp-topology" {
    uses isis-topology-type;
}

augment "/nt:network-topology/nt:topology/igp:igp-topology-attributes" {
    when "../nt:topology-types/l3t:l3-unicast-igp-topology/isis";
    container isis-topology-attributes {
        leaf net {
            type iso-net-id;
        }
    }
}

augment "/nt:network-topology/nt:topology/nt:node/igp:igp-node-attributes"
{
    when "../..nt:topology-types/l3t:l3-unicast-igp-topology/isis";
    uses isis-node-attributes;
}

augment "/nt:network-topology/nt:topology/nt:link/igp:igp-link-attributes"
{

```

```

        when "../../../nt:topology-types/l3t:l3-unicast-igp-topology/isis";
        uses isis-link-attributes;
    }

    grouping isis-node-attributes {
        container isis-node-attributes {
            container iso {
                leaf iso-system-id {
                    type iso-system-id;
                }
                leaf iso-pseudonode-id {
                    default "0";
                    type iso-pseudonode-id;
                }
            }
            leaf-list net {
                max-elements 3;
                type iso-net-id;
            }
            leaf-list multi-topology-id {
                description "List of Multi Topology Identifier upto 128 (0-127
). RFC 4915";
                max-elements "128";
                type uint8 {
                    range "0..127";
                }
            }
            choice router-type {
                case level-2 {
                    leaf level-2 {
                        type empty;
                    }
                }
                case level-1 {
                    leaf level-1 {
                        type empty;
                    }
                }
                case level-1-2 {
                    leaf level-1-2 {
                        type empty;
                    }
                }
            }
            container ted {
                uses ted:ted-node-attributes;
            }
        }
    }
}

```

```
    grouping isis-link-attributes {
      container isis-link-attributes {
        leaf multi-topology-id {
          type uint8 {
            range "0..127";
          }
        }
        container ted {
          uses ted:ted-link-attributes;
        }
      }
    }

    augment "/igp:igp-node-event" {
      uses isis-topology-type;
      uses isis-node-attributes;
    }

    augment "/igp:igp-link-event" {
      uses isis-topology-type;
      uses isis-link-attributes;
    }
  } // Module isis-topology

<CODE ENDS>
```

8. TED YANG Module

```
<CODE BEGINS>
file "ted@2013-10-21.yang"
module ted {
  yang-version 1;
  namespace "urn:TBD:params:xml:ns:yang:network:ted";
  // replace with IANA namespace when assigned
  prefix ted;

  import ietf-inet-types {
    prefix inet;
  }

  organization "TBD";
  contact
    "TBD";
  description
    "Helper module to hold TED attributes for OSPF/ISIS";

  revision 2013-10-21 {
```

```
    description
      "Initial revision";
  }

  typedef switching-capabilities {
    description
      "Switching Capabilities of an interface.";
    reference
      "RFC 5307: IS-IS Extensions in Support of Generalized
      Multi-Protocol Label Switching (GMPLS)";
    type enumeration {
      enum "PSC-1" {
        description
          "Packet-Switch Capable-1 (PSC-1)";
        value 1;
      }
      enum "PSC-2" {
        description
          "Packet-Switch Capable-2 (PSC-2)";
        value 2;
      }
      enum "PSC-3" {
        description
          "Packet-Switch Capable-3 (PSC-3)";
        value 3;
      }
      enum "PSC-4" {
        description
          "Packet-Switch Capable-4 (PSC-4)";
        value 4;
      }
      enum "L2SC" {
        description
          "Layer-2 Switch Capable (L2SC)";
        value 51;
      }
      enum "TDM" {
        description
          "Time-Division-Multiplex Capable (TDM)";
        value 100;
      }
      enum "LSC" {
        description
          "Lambda-Switch Capable (LSC)";
        value 150;
      }
      enum "FSC" {
        description
```



```
        "Fiber-Switch Capable (FSC)";
        value 200;
    }
}

typedef pcc-capabilities {
    description
        "Path Computation Capabilities.";
    reference
        "RFC 5088, draft-ietf-pce-disco-protoc-isis-07.txt
        OSPF/ISIS Protocol Extensions for Path Computation Element (PCE) Discov
ery.";
    type bits {
        bit path-computation-with-gmpls-link-constraints {
            position 0;
        }
        bit bidirectional-path-computation {
            position 1;
        }
        bit diverse-path-computation {
            position 2;
        }
        bit load-balanced-path-computation {
            position 3;
        }
        bit synchronized-path-computation {
            position 4;
        }
        bit support-for-multiple-objective-functions {
            position 5;
        }
        bit support-for-additive-path-constraints {
            position 6;
        }
        bit support-for-request-prioritization {
            position 7;
        }
        bit support-for-multiple-requests-per-message {
            position 8;
        }
    }
}

grouping ted-node-attributes {
    description
        "Identifier to uniquely identify a node in TED";
    reference "RFC 5305, RFC 6119: IPv6 Traffic Engineering in IS-IS/OSPF";
    leaf te-router-id-ipv4 {
```

```
        description
            "Globally unique IPv4 Traffic Engineering Router ID.";
        type inet:ipv4-address;
    }
    leaf te-router-id-ipv6 {
        description
            "Globally unique IPv6 Traffic Engineering Router ID";
        type inet:ipv6-address;
    }
    list ipv4-local-address {
        description
            "List of IPv4 Local Address(OSPF). RFC 5786";
        key "ipv4-prefix";
        leaf ipv4-prefix {
            description
                "Local IPv4 address for the node";
            type inet:ipv4-prefix;
        }
    }
    list ipv6-local-address {
        description
            "List of IPv6 Local Address.";
        reference
            "RFC 5786: Advertising a Router's Local Addresses
            in OSPF Traffic Engineering (TE) Extensions";
        key "ipv6-prefix";
        leaf ipv6-prefix {
            description
                "Local IPv6 address for the node";
            type inet:ipv6-prefix;
        }
        leaf prefix-option {
            description
                "IPv6 prefix option.";
            type uint8;
        }
    }
    leaf pcc-capabilities {
        description
            "OSPF/ISIS PCC capabilities";
        type pcc-capabilities;
    }
}

grouping ted-link-attributes {
    description
        "TED Attributes associated with the link.";
    reference "RFC 3630, RFC 3784: IS-IS / OSPF Traffic Engineering (TE)";
```

```

    leaf color {
      description
        "Administrative group or color of the link";
      type uint32;
    }
    leaf max-link-bandwidth {
      description
        "Maximum bandwidth that can be see on this link in this direction. Units in bytes per second";
      type decimal64 {
        fraction-digits 2;
      }
    }
    leaf max-resv-link-bandwidth {
      description
        "Maximum amount of bandwidth that can be reserved in this direction in this link. Units in bytes per second";
      type decimal64 {
        fraction-digits 2;
      }
    }
    list unreserved-bandwidth {
      description
        "Unreserved bandwidth for 0-7 priority levels. Units in bytes per second";
      max-elements "8";
      key "priority";
      leaf priority {
        type uint8 {
          range "0..7";
        }
      }
      leaf bandwidth {
        description
          "Unreserved bandwidth for this level";
        type decimal64 {
          fraction-digits 2;
        }
      }
    }
    leaf te-default-metric {
      description
        "Traffic Engineering Metric";
      type uint32;
    }
    container srlg {
      description
        "Shared Risk Link Group Attributes";
      uses srlg-attributes;
    }
  }
}

```

```
grouping srlg-attributes {
  description
    "Shared Risk Link Group Attributes";
  reference
    "RFC 5307, RFC 4203: ISIS / OSPF Extensions in Support of
    Generalized Multi-Protocol Label Switching (GMPLS)";
  list interface-switching-capabilities {
    description
      "List of interface capabilities for this interface";
    key "switching-capability";
    leaf switching-capability {
      description
        "Switching Capability for this interface";
      type ted:switching-capabilities;
    }
    leaf encoding {
      description
        "Encoding supported by this interface";
      type uint8;
    }
    list max-lsp-bandwidth {
      description
        "Maximum LSP Bandwidth at priorities 0-7";
      max-elements "8";
      key "priority";
      leaf priority {
        type uint8 {
          range "0..7";
        }
      }
      leaf bandwidth {
        description
          "Max LSP Bandwidth for this level";
        type decimal64 {
          fraction-digits 2;
        }
      }
    }
  }
  container packet-switch-capable {
    when "../switching-capability = PSC-1 or ../switching-capability = PSC
-2 or ../switching-capability = PSC-3 or ../switching-capability = PSC-4";
    description
      "Interface has packet-switching capabilities";
    leaf minimum-lsp-bandwidth {
      description
        "Minimum LSP Bandwidth. Units in bytes per second";
      type decimal64 {
        fraction-digits 2;
      }
    }
  }
}
```

```
    }
    leaf interface-mtu {
      description
        "Interface MTU";
      type uint16;
    }
  }
  container time-division-multiplex-capable {
    when "../switching-capability = TDM";
    description
      "Interface has time-division multiplex capabilities";
    leaf minimum-lsp-bandwidth {
      description
        "Minimum LSP Bandwidth. Units in bytes per second";
      type decimal64 {
        fraction-digits 2;
      }
    }
    leaf indication {
      description
        "Indication whether the interface supports Standard or Arbitrary S
ONET/SDH";
      type uint16;
    }
  }
}
list srlg-values {
  description
    "List of Shared Risk Link Group this interface belongs to.";
  key "srlg-value";
  leaf srlg-value {
    description
      "Shared Risk Link Group value";
    type uint32;
  }
}
leaf link-protection-type {
  description
    "Link Protection Type desired for this link";
  type uint16;
}
}
}

<CODE ENDS>
```

9. Security Considerations

The transport protocol used for sending the topology data MUST support authentication and SHOULD support encryption. The data-model by itself does not create any security implications.

10. Contributors

The model presented in this paper was contributed to by more people than can be listed on the author list. Additional contributors include:

- o Ken Gray, Juniper Networks
- o Tom Nadeau, Juniper Networks
- o Aleksandr Zhdankin, Cisco

11. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Ladislav Lhotka, Andy Bierman, Carlos Pignataro, and Juergen Schoenwaelder.

12. References

12.1. Normative References

- [RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, December 1990.
- [RFC2178] Moy, J., "OSPF Version 2", RFC 2178, July 1997.
- [RFC6020] Bjorklund, M., "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.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

12.2. Informative References

- [I-D.bierman-netconf-restconf] Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando, "RESTCONF Protocol", draft-bierman-netconf-restconf-02 (work in progress), October 2013.

[I-D.ietf-netmod-interfaces-cfg]
Bjorklund, M., "A YANG Data Model for Interface
Management", draft-ietf-netmod-interfaces-cfg-12 (work in
progress), July 2013.

[I-D.lhotka-netmod-yang-json]
Lhotka, L., "Modeling JSON Text with YANG", draft-lhotka-
netmod-yang-json-02 (work in progress), September 2013.

Authors' Addresses

Alexander Clemm
Cisco

EMail: alex@cisco.com

Hariharan Ananthakrishnan
Juniper Networks

EMail: hanantha@juniper.net

Jan Medved
Cisco

EMail: jmedved@cisco.com

Tony Tkacik
Cisco

EMail: ttkacik@cisco.com

Robert Varga
Pantheon Technologies SRO

EMail: robert.varga@pantheon.sk

Nitin Bahadur
Juniper Networks

EMail: nitinb@juniper.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 08, 2014

L. Huang
A. Clemm
Cisco Systems
A. Bierman
YumaWorks
September 04, 2013

YANG Data Model for Stateless Packet Filter Configuration
draft-huang-netmod-acl-03.txt

Abstract

A Stateless Packet Filter (SPF) determines which packets are allowed to transit a system according to a set of rules, applying special actions to packets as necessary. This document defines a YANG data model for the configuration of Stateless Packet Filters on 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 March 08, 2014.

Copyright Notice

Copyright (c) 2013 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	3
2. Definitions and Acronyms	4
3. The Design of the Stateless Packet Filter Data Model	5
3.1. Overall Model Structure	5
3.2. Data hierarchy	6
3.3. Other Considerations	9
3.3.1. Extensibility	9
3.3.2. SPF Chain Support	9
3.3.3. SPF Test Extensions	10
3.3.4. Attaching SPFs to interfaces	11
4. stateless-pf Module	11
4.1. Features	11
4.2. Types	12
4.3. Groupings	13
4.4. Containers	13
4.4.1. spfs Container	13
4.4.2. port-groups Container	14
4.4.3. timerange-groups Container	14
4.4.4. ip-address-groups Container	15
5. spf-ip module	16
5.1. Groupings	16
5.1.1. IP-SOURCE-NETWORK grouping	16
5.1.2. IP-DESTINATION-NETWORK grouping	17
5.1.3. DSCP-OR-TOS Grouping	17

5.1.4. IP-PFE-FILTERS Grouping	18
5.2. augment	20
5.2.1. global-fragments leaf	21
6. spf-mac module	23
6.1. MAC-SOURCE-NETWORK grouping	23
6.2. MAC-DESTINATION-NETWORK grouping	24
6.3. augment	25
7. spf-arp module	25
7.1. augment	25
8. Data Model Structure	25
9. SPF Examples	33
9.1. Configuration Example	33
10. Stateless-PF YANG Module	35
11. SPF-IP YANG Module	48
12. SPF-MAC Configuration YANG Module	62
13. SPF-ARP Configuration YANG Module	68
14. COMMON-TYPES YANG Module	71
15. Security Considerations	79
16. Open items from the previous revision	79
17. Acknowledgements	80
18. References	80
18.1. Normative References	80
18.2. Informative References	80

1. Introduction

This document defines a YANG [RFC6020] data model for the configuration of Stateless Packet Filters (SPF).

A Stateless Packet Filter is a function that filters traffic on a network device according to an ordered set of rules that define which packets are to be permitted and which are to be denied. Each rule is represented by a Packet Filter Entry (PFE). The sets of rules are sometimes also referred to as "Access Control Lists" (ACL), the rules as "Access Control Entries" (ACE) or simply "firewall rules". For the purposes of this document, we will use the terms SPF, stateless-pf and ACL interchangeably, as well as the terms PFE and ACE.

A PFE consists of two parts:

- o A set of filters with a set of matching criteria that a packet must satisfy for the rule to be applied.
- o A set of actions (most commonly, a single action) that specifies what to do with the packet when the matching criteria is met, for example, to drop the packet.

There are different types of SPF, depending on which types of packets they filter. Three of the most common types are covered in this specification: MAC SPF, IP SPF, and ARP SPF.

- o MAC SPFs: MAC SPFs are used to filter traffic using the information in the Layer 2 header of each packet. MAC SPFs are by default only applied to non-IP traffic; however, Layer 2 interfaces can be configured to apply MAC SPFs to all traffic.
- o IP SPFs: IP SPFs are ordered sets of rules that can use to filter traffic based on IP information in the Layer 3 header of packets. The device applies IP SPFs only to IP traffic. IP SPF can be IPv4 or IPv6.
- o ARP SPFs: ARP SPFs are used to filter Address Resolution Protocol (ARP) traffic.

Not every device implements every type of SPF. The model for each SPF type is therefore specified in its own YANG module. A device will implement only the modules for the SPF types that it supports. In addition, device implementations may vary greatly in terms of the filter constructs that they support for any given SPF type. Therefore, SPF YANG Module makes extensive use of the "feature" construct which allows implementations to support those SPF configuration features that lie within their capabilities.

The model can accommodate other SPF types beyond the ones that are defined in this document. For this purpose, new SPF types can be defined in their own modules which extend and augment the generic portion of the model according to the same design pattern. This way, the model serves as a framework that can be applied for any type of Stateless Packet Filter.

2. Definitions and Acronyms

AFI: Address Field Identifier

ARP: Address Resolution Protocol

CoS: Class of Service

DSCP: Differentiated Services Code Point

ICMP: Internet Control Message Protocol

IGMP: Internet Group Management Protocol

IP: Internet Protocol

IPv4: Internet Protocol version 4

IPv6: Internet Protocol version 6

MAC: Media Access Control

PFE: Packet Filter Entry

QoS: Quality of Service

SPF: Stateless Packet Filter

TCP: Transmission Control Protocol

ToS: Type of Service

TTL: Time To Live

UDP: User Datagram Protocol

VLAN: Virtual Local Area Network

VRF: Virtual Routing and Forwarding

3. The Design of the Stateless Packet Filter Data Model

3.1. Overall Model Structure

The stateless-pf data model consists of five YANG modules. The first module, "stateless-pf", defines generic SPF aspects which are common to all SPFs regardless of their type, as well as a set of auxiliary definitions. In effect, the module can be viewed as providing a generic SPF "superclass".

Three other modules, "spf-ip", "spf-mac", and "spf-arp", augment the "stateless-pf" module with definitions that are specific to different types of SPFs, specifically, SPFs for IP, MAC, and ARP, respectively. These specifics are for the largest part reflected in the Packet Filter Entries, that is, the rules which specify the filter criteria that a packet must meet for the rule to be applied, and the actions that are to be taken in case the filter matches. Keeping the modules separate provides for a more modular data model than would be the case if all types were combined into a single monolithic module.

To extend the model with other SPF types, additional modules that augment the "stateless-pf" module can be defined, thus reflecting the same model structure and following the same design pattern.

Finally, module "common-types" defines types that are used in the stateless-pf data model but are not really specific to SPF's. These definitions could potentially be of interest to other models as well; keeping them in a separate module allows to import these definitions independent of the support for SPF's.

3.2. Data hierarchy

The data hierarchy that is defined by the spf module is depicted in the following Figure "SPF Model Structure", where brackets enclose list keys, "rw" means configuration, "ro" means operational state data, and "?" means optional node. Parentheses enclose choice and case nodes. The structure is a collapsed structure and does not depict all definitions; it is intended to illustrate the overall structure. A fully expanded structure can be found in Data Model Structure Section (Section 8).

```

module: stateless-pf
  +--rw spfs
    +--rw spf [name]
      |   +--rw name
      |   +--rw spf-type
      |   +--rw enable-capture-global?
      |   +--rw capture-session-id-global?
      |   +--rw (enable-match-counter-choices)?
      |   +--ro match?
      |
    +--rw port-groups
      |   +--rw port-group [name]
      |   |   +--rw name
      |   |   +--rw port-group-entry
      |   +--rw timerange-groups
      |   |   +--rw timerange-group [name]
      |   |   |   +--rw name
      |   |   |   +--rw time-range
      |   +--rw ip-address-groups
      |   |   +--rw ip-address-group [name]
      |   |   |   +--rw name
      |   |   |   +--rw afi?
      |   |   |   +--rw ip-address

```

SPF Model Structure

Data nodes in the stateless-spf module are contained under a single container node, "spfs". This node contains a list, "spf". Each SPF is represented by an element in that list and identified by a name

that serves as key to the list. Interfaces (which are not part of the model, but for example defined per [if-config]) to which an SPF is applied can then refer to the SPF using that name, respectively a data type "spf-ref" introduced for that purpose. Each spf list element has furthermore a type, as indicated through "spf-type". The spf-type determines which types of PFEs can be contained in an SPF. The PFE definitions themselves are provided by the spf-ip, spf-mac, and spf-arp modules, which augment the spf definition in the spf module accordingly. The subsequent data nodes in the spf list allow to configure whether packets that match an SPF should be captured for further analysis. Finally, the list contains an object that maintains a counter of the number of SPF matches.

Auxiliary objects "port-groups", "ip-address-groups", "timerange-groups" are used to define groupings of ports and of IP-addresses as well as schedule information, respectively. They are in effect convenience objects which allow PFEs to refer to groupings and schedules by name, rather than needing to re-specify them in each PFE where they apply.

The following figure depicts how different types of PFEs are inserted into that structure. As indicated earlier, the corresponding definitions are provided in separate modules that augment the spf module. In the data structure, the augmenting module is indicated by the prefix of the corresponding data nodes: "spf-ip", "spf-mac", and "spf-arp", respectively. PFEs for IPv4 and for IPv6 are both defined in the same module, spf-ip. While it would have been possible to define each in its own separate module, it was a design decision to combine them, as they share enough commonality that a separation would have resulted in a considerable amount of definition redundancy.

The figure does not depict objects not pertinent to that structure, such as objects intended to make the definition of port groups ("port-groups"), timeranges ("time-range-groups"), and IP address groups ("ip-address-groups") reusable, as well as objects that are contained in spf list elements, such as "name" and "enable-capture-global".

```

module: stateless-pf
  +--rw spfs
    +--rw spf [name]
      |   +--rw spf-ip:afi
      |   +--rw spf-ip:ipv6-pfes
      |   |   +--rw spf-ip:ipv6-pfe [name]
      |   |   +--rw spf-ip:name
      |   |   +--rw (remark-or-ipv6-case)?
      |   |   +--:(remark)

```

```

| | | +---rw spf-ip:remark
| | | +---:(ipv6-pfe)
| | | +---rw spf-ip:filters
| | | +--- filter parameters
| | | +---rw spf-ip:actions
| | | +--- action parameters
| | | +--- ro spf-ip:match

module: stateless-pf
+---rw spfs
+---rw spf [name]
| +---rw spf-ip:afi
| +---rw spf-ip:ipv4-pfes
| | +---rw spf-ip:ipv4-pfe [name]
| | +---rw spf-ip:name
| | +---rw (remark-or-ipv4-pfe)?
| | +---:(remark)
| | | +---rw spf-ip:remark
| | | +---:(ipv4-pfe)
| | | +---rw spf-ip:filters
| | | +--- filter parameters
| | | +---rw spf-ip:actions
| | | +--- action parameters
| | | +--- ro spf-ip:match

module: stateless-pf
+---rw spfs
+---rw spf [name]
| +---rw spf-mac:mac-pfes
| | +---rw spf-mac:mac-pfe [name]
| | +---rw spf-mac:name
| | +---rw (remark-or-mac-pfe)?
| | +---:(remark)
| | | +---rw spf-mac:remark
| | | +---:(mac-pfe)
| | | +---rw spf-mac:filters
| | | +--- filter parameters
| | | +---rw spf-mac:actions
| | | +--- action parameters
| | | +--- ro spf-mac:match

module: stateless-pf
+---rw spfs
+---rw spf [name]
| +---rw spf-arp:arp-pfes
| | +---rw spf-arp:arp-pfe [name]

```

```

| | +---rw spf-arp:name
| | +---rw (remark-or-arp-pfe)?
| |   +---:(remark)
| |     +---rw spf-arp:remark
| |   +---:(arp-pfe)
| |     +---rw spf-arp:filters
| |       +--- filter parameters
| |     +---rw spf-arp:actions
| |       +--- action parameters
| |   +--- ro spf-arp:match

```

Model structure - different SPF types

As is evident from Figure "Model structure - different SPF types", the same generic design pattern is reflected in every SPF type. Each SPF contains a list of PFEs, identified by a name by which PFEs in the list are ordered. Each PFE consists either of a remark or of an actual access control rule. Remarks are in effect comment lines inside an SPF that are intended for human or administrator consumption. They are included in the YANG module to maintain consistency with CLI. Access control rules, on the other hand, consist of a left hand side ("filters") that specifies a set of matching criteria and a right hand side ("actions") that specifies the action to take when matching criteria are met. An overview of the full list of filter and parameters is given in Section 8.

Since the design pattern for each SPF type is the same, an alternative design to the YANG modules would have been to extend the "spf" module to include the data nodes up to the level depicted in Figure "Model structure - different SPF types", as the real distinction occurs in the filter and action parameters that occur below it. In that case, however, the corresponding data nodes would have had to contend with more complex conditions. The modules defined here aim at keeping complexity of definitions within the modules as low as possible, at the price of repeating a few data nodes that provide the overall top level structure.

3.3. Other Considerations

3.3.1. Extensibility

If needed, the model can be extended for other types of SPFs in straightforward manner. New types of SPFs can be defined in additional YANG modules that apply the same design patterns much in the same way as in the case of IP, MAC, and ARP SPFs.

3.3.2. SPF Chain Support

SPF chains are used in some application domains. SPF chains are not included in the data model, but could be accommodated in the model through extensions in a straightforward way.

SPF chains work roughly as follows. In an SPF chain, as an alternative to an action, an PFE can point to another SPF. If a packet matches the filter condition, it is subjected to the other SPF. If the other SPF contains an PFE that matches, that action is executed. If there is no match, processing is returned to the first SPF and processing continues with the subsequent PFEs until a match is found. This way, chained SPFs can be considered as a special form of "SPF subroutine".

An example of an SPF chain might be a rule that contains a filter for a specific destination port number in an IP packet, then invokes another SPF that contains a specific set of firewall rules for traffic directed at that particular port. Even though the data model for SPF presented in this document uses a flat list of PFE in each SPF, the actions in the model can be augmented to support SPF chains.

The model can be extended with SPF chains roughly as follows: A new spf-chaining action is introduced, represented as a leaf whose value contains a reference to an SPF as a parameter. Below is an example of how the spf-ip model could be extended to support SPF chains for ip-v4:

```
augment "/spf:spf/spf:spf-ip:ipv4-pfes" +
  "/spf-ip:ipv4-pfe/spf-ip:actions" {
    leaf chain {
      type spf-ref ;
      description "Reference to another SPF name to chain the PFEs";
    }
  }
```

For SPFs that are expected to not terminate when no PFE matches, but return processing to the invoking SPF, an optional SPF parameter can be introduced that indicates for chained SPFs which chaining behavior should apply.

3.3.3. SPF Test Extensions

Given the complexity of SPFs in many deployments, debugging SPFs and assessing whether an SPF has the actual desired effect can be a challenge. In order to facilitate those tasks and allow to check whether an SPF has indeed the intended effect, an additional

administrative function that allows applications and users to test a packet against the SPF can be introduced. The function can take the form of an RPC which takes as input parameter a leaf with the reference to the SPF that is to be tested, and a leaf with a packet. The output parameter includes a leaf indicating the action that is taken as a result, as well as a leaf with the reference to the matching PFE.

3.3.4. Attaching SPFs to interfaces

SPFs typically do not exist in isolation. Instead, they are associated with a certain scope in which they are applied, for example, an interface of a set of interfaces. How to attach an SPF to an interface (or other system artifact) is outside the scope of this model, as it depends on the specifics of the system model that is being applied. However, in general, the general design pattern will involve adding a data node with a reference, or set of references, to SPFs that are to be applied to the interface. For this purpose, the type definition "spf-ref" can be used.

For example, to attach an SPF to an interface as defined per the data model [if-config], the following steps can be applied:

- o Introduce a new YANG module to extend the interface configuration YANG module.
- o Import modules "interfaces" [if-config] (prefix: "if") and "stateless-pf" (prefix: "spf").
- o Augment list "interface" (/if:interfaces/if:interface) with a leaf-list of type "spf:spf-ref".

4. stateless-pf Module

Module "stateless-pf" is a top container module for all SPFs. It contains a container "spfs" with a list "spf" of named SPFs. Modules "spf-ip", "spf-mac", and "spf-arp" augment this list with the objects that are specific to each respective type of SPF. In addition, module "spf" also defines a set of features, reusable types, and reusable groupings.

4.1. Features

When it comes to SPF implementations, a wide range of different capabilities exists across devices. For example, not every device implements every type of SPF. Some devices may support time-based SPFs that are only in effect during specified times, others may not.

In order to accommodate this wide range of capabilities, this data model makes extensive use of the "feature" construct. The defined features allow implementations to declare which capabilities they support, and only support the corresponding portions of the data model.

4.2. Types

The definition of SPF's requires a number of new data types introduced in this data model. Table 1 depicts data types that are unique to SPF's. Table 2 depicts data types that are required by SPF's, but not specific to them, and that may hence be reused by other models. Those data types are defined in module "common-types". For details of each type, please refer to the corresponding typedef descriptions and references in the model.

YANG type	base type
spf-comparator	enumeration
spf-action	enumeration
spf-remark	string
spf-type-ref	identityref
spf-ref	leafref
port-group-ref	leafref
ip-address-group-ref	leafref
time-range-Ref	leafref
weekdays	bits
spf-name-string	string

Table 1

YANG type	base type
cos	uint8
tos	uint8
precedence	uint8
tcp-flag-type	enumeration
ether-type	string
ip-protocol	uint8
igmp-code	uint8
icmp-type	uint32
icmp-code	uint32
vlan-identifier	uint16
time-to-live	uint32

+-----+-----+

Table 2

4.3. Groupings

The data model defines two groupings, PFE-COMMON and FILTER-COMMON.

- o PFE-COMMON is a collection of nodes that should be added to every PFE list entry. PFE-COMMON contains the actions container and a read-only match leaf. The actions container contains two leaves.
 - * An "action" leaf that specifies what to do with the packet when the matching criteria is met, for example, to drop the packet.
 - * A "log" leaf that indicates whether to create a log entry when an pfe filter matches. (Some devices may not support a log capability. Hence support of this leaf is conditional on declaration of a corresponding feature, as indicated by use of the "if-feature" construct.)
- o FILTER-COMMON is a collection of nodes that should be added to every 'filters' container within each PFE list entry.

4.4. Containers

4.4.1. spfs Container

Container "spfs" contains a list "spf" of named SPFs. Each list element "spf" contains the following global leaves. The list elements are augmented with additional data nodes defined in modules "spf-arp", "spf-mac", and "spf-ip".

- o name
- o spf-type
- o enable-capture-global
- o capture-session-id-global
- o enable-match-counter-choices: The difference of these two choices is that "enable-match-counter" indicates to collect total match statistics for all pfes, whereas "enable-per-entry-match-counter" indicates to collect match statistics for each PFE.
- o match

4.4.2. port-groups Container

Container "port-groups" allows to classifying protocol port into groups. It contains a sequence of "port-group" data nodes. Each "port-group" defines a range of ports and can be referred to by name. Multiple PFEs can refer to the same port group. The following is a Netconf XML example of port-groups and how it is referred to from an PFE.

```
<src-port-group-name>
<port-group-name>port-tunnell</port-group>
</src-port-group-name>

<port-groups>
  <port-group>
    <name>port-tunnell</name>
    <port-group-entry>
      <name>http-proxy</name>
      <port-lower>21</port-lower>
      <port-upper> 22</port-upper>
    </port-group-entry>
  </port-group>
</port-groups>
```

4.4.3. timerange-groups Container

Container "timerange-groups" container contains a list, "timerange-group". Each of its elements defines a sequence of time ranges, "time-range". Each time-range object consists of either a remark (comments for the time range), or of an absolute time for start or end (or both) of the time range, or a periodic time for start or end or both. Object "remark" contains administrator-provided comments for the time-range that will be kept in the device. Like with port groups, the same time-range can be reused by different PFEs. The following is a Netconf XML example of a timerange group that contains a remark and a single time range.

```
<timerange-groups>
  <timerange-group>
    <name>weekday</name>
    <time-range>
      <name>10</name>
      <remark> email server maintenance</remark>
    </time-range>
    <time-range>
      <name>20</name>
```

```
<periodic>
  <weekday>
    Monday Tuesday Wednesday Thursday Friday
  </weekday>
  <start> 21:00:00</start>
  <end> 24:00:00</end>
</periodic>
</time-range>
</timerange-group>
</timerange-groups>
```

4.4.4. ip-address-groups Container

Container "ip-address-groups" contains is list "ip-address-group" of named IP address groups. Each IP address group is a sequence of pairs "ip-address" and "mask", or a pair of "host" and "host-address". Each IP address group can be referred from an PFE by name. The following is a Netconf XML example of an IP address group and how it is referred to from an PFE.

```
<ip-address-groups>

  <ip-address-group>
    <name>Email-Server-IPV4</name>
    <ip-addresses>
      <ip-address>
        <name>10</name>
        <ip-address>128.107.0,0</ip-address>
        <ip-mask>255.255.0.0</ip-mask>
      </ip-address>
      <ip-address>
        <name>20</name>
        <ip-address>139.207.0.0</ip-address>
        <ip-mask>255.255.0.0</ip-mask>
      </ip-address>
    </ip-addresses>
  </ip-address-group>
</ip-address-groups>

<ip-pfe>
  <name>100</name>
  <afi>ipv4</afi>
  <actions>permit</actions>
  <filters>
    <ip-source-group>Email-Server-IPV4</ip-source-group>
    <ip-dest-any/>
```

```

    </filters>
</ip-pfe>

```

5. spf-ip module

spf-ip is the module that defines IP-SPF. It augments spf list in spf module.

5.1. Groupings

5.1.1. IP-SOURCE-NETWORK grouping

```

IP-SOURCE-NETWORK
  +---rw (source-address-host-group)?
  +---:(source-ip)
  |   +---rw ip-source-address      inet:ip-address
  |   +---rw ip-source-mask        inet:ip-address
  +---:(ip-source-any)
  |   +---rw ip-source-any         empty
  +---:(source-host)
  |   +---:(ip-src-host-address-or-name)
  |   |   +---:(ip-source-host-address)
  |   |   |   +---rw ip-source-host-address      inet:ip-address
  |   |   +---:(ip-source-host-name)
  |   |   |   +---rw ip-source-host-name         inet:domain-name
  +---:(source-group)
  |   +---rw ip-source-group?       ip-address-group-ref

```

IP-SOURCE-NETWORK is a reusable grouping. It allows five ways to specify a network: ip with mask, any network, host-name or host address, reference to a predefined ip address group. Here are valid example instances:

o ip with mask:

```

<ip-source-address>192.168.1.0</ip-source-address>
<ip-source-mask>255.255.255.0</ip-source-mask>

```

o any network:

```

<ip-source-any/>

```

o host-name:

```
<ip-source-host-name>switch1</ip-source-host-name>
```

- o host-address:

```
<ip-source-host-address>192.168.1.2</ip-source-host-address>
```

- o reference to a predefined ip address group (Email-Server-IPV4 is defined in Section 4.4.4):

```
<ip-source-group>Email-Server-IPV4</ip-source-group>
```

5.1.2. IP-DESTINATION-NETWORK grouping

```
IP-DESTINATION-NETWORK
+--rw (dest-address-host-group)?
+--:(dest-ip)
|   +--rw ip-dest-address      inet:ip-address
|   +--rw ip-dest-mask?       inet:ip-address
+--:(ip-dest-any)
|   +--rw ip-dest-any         empty
+--:(dest-host)
|   +--:(ip-dest-host-address-or-name)
|   |   +--:(ip-dest-host-address)
|   |   |   +--rw ip-dest-host-address      inet:ip-address
|   |   +--:(ip-dest-host-name)
|   |   |   +--rw ip-dest-host-name         inet:domain-name
+--:(group)
|   +--rw ip-dest-group?       ip-address-group-ref
```

IP-DESTINATION-ADDRESS is a reusable grouping. Its structure is similar to IP-SOURCE-NETWORK. The reason to have both IP-SOURCE-NETWORK and IP-DESTINATION-NETWORK groupings is to allow "ip-source-address" and "ip-destination-address" leaves to appear in the same container. For example:

```
<filters>
  <ip-source-address>192.168.1.0</ip-source-address>
  <ip-source-mask>255.255.255.0</ip-source-mask>
  <ip-dest-address>any</ip-dest-address>
</filters>
```

5.1.3. DSCP-OR-TOS Grouping

DSCP-OR-TOS grouping defines a choice, "dscp-or-tos". It allows two ways to filter for a QoS packet:

- o dscp: Match packet on DSCP value.
- o tos: Match packet on TOS and precedence value.

The typedef for "tos" and "precedence" is defined in module "common-types", which could be deprecated should IETF define a separate set of definitions.

5.1.4. IP-PFE-FILTERS Grouping

```

IP-PFE-FILTERS
  +--rw protocol?                               c-types:ip-protocol
  +--spf:FILTER-COMMON
  +--rw fragments?                             empty
  +--rw time-range?                           spf:Time-Range-Ref
  +-- (src-ports)?
    | +--rw (port-number-or-range)?
    | | +--:(port-number-range)
    | | | +--rw src-port-lower?             inet:port-number
    | | | +--rw src-port-upper?            inet:port-number
    | | +--:(port-number)
    | | | +--rw src-comparator               comparator
    | | | +--rw src-port?                   inet:port-number
    | | +--:(port-group-ref)
    | | | +--src-port-group-name
    +-- (des-ports)?
    | +--rw (port-number-or-range)?
    | | +--:(port-number-range)
    | | | +--rw des-port-lower?             inet:port-number
    | | | +--rw des-port-upper?            inet:port-number
    | | +--:(port-number)
    | | | +--rw des-comparator               comparator
    | | | +--rw des-port?                   inet:port-number
    | | +--:(by-name)
    | | | +-- des-port-group-name
  +--rw icmp-type?                             c-types:icmp-type
  +--rw icmp-code?                             c-types:icmp-type
  +--rw (packet-length-or-range)?
    | +--:(length)
    | | +--rw packet-length-comparator      spf:Comparator
    | | +--rw packet-length                 uint32
    | | +--:(range)
    | | | +--rw packet-length-upper          uint32
    | | | +--rw packet-length-lower          uint32
  +--rw tcp-flag-value?                       c-types:tcp-flag-type

```

```

+---rw tcp-flag-mask?                c-types:tcp-flag-type
+---rw tcp-flag-operation?           enumeration
+---rw (ttl-value-or-range)?
    +---:(value)
        |   +---rw ttl-comparator?    spf:spf-comparator
        |   +---rw ttl-value?         c-types:Time-to-Live
    +---:(range)
        +---rw ttl-value-lower?       c-types:Time-to-Live
        +---rw :ttl-value--upper?    c-types:Time-to-Live

```

IP-PFE-FILTERS defines the following leaves that are used by both by IPv4 and IPv6 PFEs:

- o protocol
- o spf:FILTER-COMMON: see Section 4.3
- o fragments: When present, it matches the non-initial fragment.
- o time-range: Enable packet capture on this filter for a timerange-group by name. time-range is Time-Range-Ref type which is a leafref.
- o src-ports choice: Allows the following three ways to define a group of ports.
 - * port-number-range: Use "src-port-lower" and "src-port-upper" leaves to specify a port range. The value of "src-port-lower" has to be less than or equal the value of "src-port-upper".
 - * port-number: Use "comparator" and "src-port" leaves to specify a port range. See Comparator typedef in the model for the possible values the "comparator" leaf.
 - * port range ref: Refer to a named port group that is defined using port-groups. For example:


```
<port-group-name>port-tunnell</port-group-name>
```
- o dest-ports choice: Analogous to "src-ports".
- o packet-length-or-range: Allows two ways to specify packet length range.

case length: Use comparator and a single packet-length to specify the range.

case range: Use packet-length-lower and packet-length-upper to specify a range. The value of packet-length-lower must be lower than or equal to the value of packet-length-upper.

- o icmp-type
- o icmp-code
- o packet-length-or-range choice
- o tcp-flag-value: tcp-flag-value, tcp-flag-mask and tcp-flag-operation allow to match any combination of packet tcp flag values.

The following example is to match the packet tcp flag ack=1, syn=1, and fin=0;

```
<tcp-flag-value> ack syn <tcp-flag-value>
<tcp-flag-mask>ack syn fin</tcp-flag-mask>
<tcp-flag-operation>match-all</tcp-flag-operation>
```

- o tcp-flag-mask
- o tcp-flag-operation
- o ttl-value-or-range

5.2. augment

The module "spf-ip" augments the definition of data node "/spf:spfs/spf:spf" with additional leaves and subcomponents.

- o afi
- o ipv6-pfes: It contains a list of ipv6-pfe. Each ipv6-pfe is either a remark or a real access control filters. The case ipv6-pfe defines the filters and actions for ipv6-pfe. The pfe uses filters defined in grouping IP-SOURCE-NETWORK, IP-DESTINATION-NETWORK, IP-PFE-FILTERS, DSCP-OR-TOS. In addition, it also allows filter on igmp-type and flow-label,
- o ipv4-pfes: ipv4-pfe has similar structure to ipv6-pfes.
- o global-fragments

5.2.1. global-fragments leaf

global-fragments is an optional leaf. It has an enumeration value of not-set, permit-all, deny-all. not-set is the default value. When the global-fragments is permit-all or deny-all, it is to permit or deny the implicit pfe fragment filter. Here is an example of implicit pfe and how the implicit pfe is affected when global-fragments is set.

Example 1: The spf configuration from the management interface with global-fragments is absent.

YANG instance of this cli configuration:

```
<spfs>
  <spf>
    <name>fragment_test1</name>
    <afi>ipv4</afi>
    <spf-type>ip-spf</spf-type>
    <ip-pfes>
      <name>10</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>192.168.5.0</ip-source-address>
        <ip-source-mask>255.255.255.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
      </filters>
    </ip-pfes>
    <ip-pfes>
      <name>20</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>189.168.0.0</ip-source-address>
        <ip-source-mask>255.255.0.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
        <fragments/>
      </filters>
    </ip-pfes>
  </spf>
</spfs>
```

By taking all the tags out, the above yang can be express in a summary of cli format like the following:

```
fragment_test1 ip-spf ipv4
10 permit ip 192.168.5.0 255.255.255.0 any
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
```

The spf configuration together with implicit pfe in the device will be:

```
fragment_test1 ip-spf ipv4
10 permit ip 192.168.5.0 255.255.255.0 any
11 permit ip 192.168.5.0 255.255.255.0 any fragment
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
100 deny any any
110 deny any any fragment
```

Notice three lines of configuration. 11, 100 and 110, are implicit.

Example 2: The spf configuration from the management interface with global-fragments

```
<spfs>
  <spf>
    <name>fragment_test2</name>
    <spf-type>ip-spf</spf-type>
    <global-fragments>deny-all</global-fragments>
    <afi>ipv4</afi>
    <ip-pfes>
      <name>10</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
        <ip-source-address>192.168.5.0</ip-source-address>
        <ip-source-mask>255.255.255.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
      </filters>
    </ip-pfes>
    <ip-pfes>
      <name>20</name>
      <actions>
        <action>permit</action>
      </actions>
      <filters>
```

```

        <ip-source-address>189.168.0.0</ip-source-address>
        <ip-source-mask>255.255.0.0</ip-source-mask>
        <ip-dest-address>any</ip-dest-address>
        <fragments/>
    </filters>
</ip-pfes>
</spf>
</spfs>

```

The spf configuration in the device with implicit aces. The deny-all void "11 permit ip 1.1.1.1/16 any fragment" pfe in previous example.

By taking all the tags out, the above yang can be express in a summary of cli format like the following:

```

fragment_test2 ip-spf ipv4 deny-all
10 permit ip 192.168.5.0 255.255.255.0 any
20 permit ip 189.168.0.0 255.255.0.0 any fragment.

```

The spf configuration together with implicit pfe in the device will be:

```

fragment_test2 ip-spf ipv4
10 permit ip 192.168.5.0 255.255.255.0 any
20 permit ip 189.168.0.0 255.255.0.0 any fragment.
100 deny any any
110 deny any any fragment

```

6. spf-mac module

6.1. MAC-SOURCE-NETWORK grouping

```

MAC-SOURCE-NETWORK
+--rw (source-network)?
+--:(source-mac)
|   +--rw source-address          yang:mac-address
|   +--rw source-address-mask     yang:mac-address
+--:(source-any)
|   +--rw source-any              empty
+--:(source-host)
|   +--rw spf-mac:source-host-name inet:host

```

MAC-SOURCE-ADDRESS is a reusable grouping. It allows to express the three kinds network.

any network: use source-any to express any network.

```
<mac-source-kind>any</mac-source-kind>
```

single host network.

```
<source-host-name>my-host</source-host-name>
```

host address with a mask.

```
<source-address>0180.c200.000</source-address>
<source-address-mask>0000.0000.0000</source-address-mask>
```

6.2. MAC-DESTINATION-NETWORK grouping

```
MAC-DESTINATION-NETWORK
  +--rw (dest-network)?
  |   +--:(address)
  |   |   +--rw dest-address          yang:mac-address
  |   |   +--rw dest-address-mask     yang:mac-address
  |   +--:(dest-any)
  |   |   +--rw dest-any              empty
  |   +--:(host)
  |       +--rw spf-mac:dest-host-name      inet:host
```

MAC-DESTINATION-ADDRESS is a reusable grouping similar to MAC-SOURCE-ADDRESS. The reason to have both MAC-SOURCE-ADDRESS and MAC-DESTINATION-ADDRESS grouping is to allow source-address and destination-address leaves appear in the same container. For example:

```
<filters>
  <source-address>0180.c200.000</source-address>
  <source-address-mask>0000.0000.0000</source-address-mask>
  <dest-any/>
</filters>
```

6.3. augment

The module "spf-mac" augments the definition of data node `"/spf:spfs/spf:spf"` with additional leaves and subcomponents. spf-mac has similar structure as spf-ipv4 and spf-ipv6 except the filters are different. mac-pfe has filters defined in grouping MAC-SOURCE-NETWORK, MAC-DESTINATION-NETWORK, spf:FILTER-COMMON, ethertype-mask, cos, time-range, and vlan.

7. spf-arp module

7.1. augment

The module "spf-arp" augments the definition of data node `"/spf:spfs/spf:spf"` with additional leaves and subcomponents.

```
augment "/spf:spfs/spf:spf"
  +--rw spf-arp:arp-pfes
    +--rw spf-arp:arp-pfe [name]
      +--rw spf-arp:name          spf:spf-name-string
      +--rw (remark-or-arp-pfe)?
        +--:(remark)
        | +--rw spf-arp:remark?    spf:spf-remark
        +--:(arp-pfe)
          +--rw filters
            +--rw direction?          enumeration
            +--spf-ip:IP-SOURCE-NETWORK
            +--spf-ip:IP-DESTINATION-NETWORK
            +--spf-mac:MAC-SOURCE-NETWORK
            +--spf-mac:MAC-DESTINATION-NETWORK
            +--spf:FILTER-COMMON
          +spf:PFE-COMMON
```

8. Data Model Structure

The combined data model for SPF configuration is structured as follows. "spf" defines the generic components of an spf system. "spf-ip", "spf-mac", "spf-arp" augment the "spf" module with additional data nodes that are needed for ip, mac, and arp spf respectively.

```
module: stateless-pf
  +--rw spfs
    +--rw spf [name]
      +--rw name
      +--rw spf-type
      +--rw enable-capture-global?
```



```

+---rw capture-session-id-global?
+---rw (enable-match-counter-choices)?
|   +---:(match)
|   |   +---rw enable-match-counter?
|   +---:(per-entry-match)
|   |   +---rw enable-per-entry-match-counter?
+---ro match?
+---rw spf-ip:afi?
+---rw spf-ip:ipv6-pfes
|   +---rw spf-ip:ipv6-pfe [name]
|   |   +---rw spf-ip:name          spf:spf-name-string
|   |   +---rw (remark-or-ipv6-case)?
|   |   |   +---:(remark)
|   |   |   |   +---rw spf-ip:remark?      spf:spf-remark
|   |   +---:(ipv6-pfe)
|   |   +---rw spf-ip:filters
|   |   |   +---rw (source-address-host-group)
|   |   |   |   +---:(source-ip)
|   |   |   |   |   +---rw spf-ip:ip-source-address
|   |   |   |   |   +---rw spf-ip:ip-source-mask
|   |   |   |   +---:(ip-source-any)
|   |   |   |   |   +---rw spf-ip:ip-source-any?
|   |   |   +---:(source-host)
|   |   |   |   +---rw (ip-src-address-or-name)
|   |   |   |   |   +---:(ip-source-host-address)
|   |   |   |   |   |   +---rw spf-ip:ip-source-host-address?
|   |   |   |   |   +---:(ip-source-host-name)
|   |   |   |   |   |   +---rw spf-ip:ip-source-host-name?
|   |   |   +---:(source-group)
|   |   |   |   +---rw spf-ip:ip-source-group?
|   |   +---rw (dest-address-host-group)
|   |   |   +---:(dest-ip)
|   |   |   |   +---rw spf-ip:ip-dest-address
|   |   |   |   +---rw spf-ip:ip-dest-mask
|   |   |   +---:(ip-dest-any)
|   |   |   |   +---rw spf-ip:ip-dest-any?
|   |   |   +---:(dest-host)
|   |   |   |   +---rw (ip-dest-address-or-name)
|   |   |   |   |   +---:(ip-dest-host-address)
|   |   |   |   |   |   +---rw spf-ip:ip-dest-host-address?
|   |   |   |   |   +---:(ip-dest-host-name)
|   |   |   |   |   |   +---rw spf-ip:ip-dest-host-name?
|   |   |   +---:(dest-group)
|   |   |   |   +---rw spf-ip:ip-dest-group?
|   +---rw spf-ip:protocol?
|   +---rw spf-ip:enable-capture?
|   +---rw spf-ip:capture-session-id?
+---rw spf-ip:fragments?

```

```

+---rw spf-ip:time-range?
+---rw (src-ports)?
|   +---:(port-number-range)
|   |   +---rw spf-ip:src-port-lower
|   |   +---rw spf-ip:src-port-upper
|   +---:(port-number)
|   |   +---rw spf-ip:src-comparator
|   |   +---rw spf-ip:src-port
|   +---:(port-group-ref)
|       +---rw spf-ip:src-port-group-name
+---rw (dest-ports)?
|   +---:(port-number-range)
|   |   +---rw spf-ip:des-port-lower
|   |   +---rw spf-ip:des-port-upper
|   +---:(port-number)
|   |   +---rw spf-ip:des-comparator
|   |   +---rw spf-ip:des-port
|   +---:(port-group-ref)
|       +---rw spf-ip:des-port-group-name
+---rw spf-ip:icmp-type?
+---rw spf-ip:icmp-code?
+---rw (packet-length-or-range)?
|   +---:(length)
|   |   +---rw spf-ip:packet-length-comparator
|   |   +---rw spf-ip:packet-length
|   +---:(range)
|       +---rw spf-ip:packet-length-upper
|       +---rw spf-ip:packet-length-lower
+---rw spf-ip:tcp-flag-value?
+---rw spf-ip:tcp-flag-mask?
+---rw spf-ip:tcp-flag-operation?
+---rw (ttl-value-or-range)?
|   +---:(value)
|   |   +---rw spf-ip:ttl-comparator?
|   |   +---rw spf-ip:ttl-value?
|   +---:(range)
|       +---rw spf-ip:ttl-value-lower?
|       +---rw spf-ip:ttl-value--upper?
+---rw (dscp-or-tos)?
|   +---:(dscp)
|   |   +---rw spf-ip:dscp?
|   +---:(tos)
|       +---rw spf-ip:tos?
|       +---rw spf-ip:precedence?
+---rw spf-ip:igmp-type?
+---rw spf-ip:flow-label?
+---rw spf-ip:actions
|   +---rw spf-ip:action

```

```

| +---rw spf-ip:log?
| +---ro spf-ip:match?
+---rw spf-ip:ipv4-pfes
| +---rw spf-ip:ipv4-pfe [name]
| +---rw spf-ip:name          spf:spf-name-string
| +---rw (remark-or-ipv4-pfe)?
| +---:(remark)
| | +---rw spf-ip:remark?      spf:spf-remark
| +---:(ipv4-pfe)
| +---rw spf-ip:filters
| | +---rw (source-address-host-group)
| | | +---:(source-ip)
| | | | +---rw spf-ip:ip-source-address
| | | | +---rw spf-ip:ip-source-mask
| | | +---:(ip-source-any)
| | | | +---rw spf-ip:ip-source-any?
| | | +---:(source-host)
| | | | +---rw (ip-src-address-or-name)
| | | | | +---:(ip-source-host-address)
| | | | | | +---rw spf-ip:ip-source-host-address?
| | | | | +---:(ip-source-host-name)
| | | | | | +---rw spf-ip:ip-source-host-name?
| | | +---:(source-group)
| | | | +---rw spf-ip:ip-source-group?
| +---rw (dest-address-host-group)
| | +---:(dest-ip)
| | | +---rw spf-ip:ip-dest-address
| | | +---rw spf-ip:ip-dest-mask
| | | +---:(ip-dest-any)
| | | | +---rw spf-ip:ip-dest-any?
| | | +---:(dest-host)
| | | | +---rw (ip-dest-address-or-name)
| | | | | +---:(ip-dest-host-address)
| | | | | | +---rw spf-ip:ip-dest-host-address?
| | | | | +---:(ip-dest-host-name)
| | | | | | +---rw spf-ip:ip-dest-host-name?
| | | +---:(dest-group)
| | | | +---rw spf-ip:ip-dest-group?
+---rw spf-ip:protocol?
+---rw spf-ip:enable-capture?
+---rw spf-ip:capture-session-id?
+---rw spf-ip:fragments?
+---rw spf-ip:time-range?
+---rw (src-ports)?
| +---:(port-number-range)
| | +---rw spf-ip:src-port-lower
| | +---rw spf-ip:src-port-upper
| +---:(port-number)

```

```

| | | | +---rw spf-ip:src-comparator
| | | | +---rw spf-ip:src-port
| | | | +---:(port-group-ref)
| | | | +---rw spf-ip:src-port-group-name
+---rw (dest-ports)?
| | | | +---:(port-number-range)
| | | | | +---rw spf-ip:des-port-lower
| | | | | +---rw spf-ip:des-port-upper
| | | | +---:(port-number)
| | | | | +---rw spf-ip:des-comparator
| | | | | +---rw spf-ip:des-port
| | | | +---:(port-group-ref)
| | | | | +---rw spf-ip:des-port-group-name
+---rw spf-ip:icmp-type?
+---rw spf-ip:icmp-code?
+---rw (packet-length-or-range)?
| | | | +---:(length)
| | | | | +---rw spf-ip:packet-length-comparator
| | | | | +---rw spf-ip:packet-length
| | | | +---:(range)
| | | | | +---rw spf-ip:packet-length-upper
| | | | | +---rw spf-ip:packet-length-lower
+---rw spf-ip:tcp-flag-value?
+---rw spf-ip:tcp-flag-mask?
+---rw spf-ip:tcp-flag-operation?
+---rw (ttl-value-or-range)?
| | | | +---:(value)
| | | | | +---rw spf-ip:ttl-comparator?
| | | | | +---rw spf-ip:ttl-value?
| | | | +---:(range)
| | | | | +---rw spf-ip:ttl-value-lower?
| | | | | +---rw spf-ip:ttl-value--upper?
+---rw (dscp-or-tos)?
| | | | +---:(dscp)
| | | | | +---rw spf-ip:dscp?
| | | | +---:(tos)
| | | | | +---rw spf-ip:tos?
| | | | | +---rw spf-ip:precedence?
+---rw spf-ip:actions
| | | | +---rw spf-ip:action      spf:spf-action
| | | | +---rw spf-ip:log?      empty
+---ro spf-ip:match?          yang:counter64
+---rw spf-ip:global-fragments? enumeration
+---rw spf-mac:mac-pfes
| | | | +---rw spf-mac:mac-pfe [name]
| | | | | +---rw spf-mac:name      spf:spf-name-string
| | | | +---rw (remark-or-mac-pfe)?
| | | | | +---:(remark)

```

```

|   +---rw spf-mac:remark?      spf:spf-remark
+---:(mac-pfe)
+---rw spf-mac:filters
|   +---rw (source-network)
|   |   +---:(source-mac)
|   |   |   +---rw spf-mac:source-address
|   |   |   +---rw spf-mac:source-address-mask
|   |   +---:(source-any)
|   |   |   +---rw spf-mac:source-any?
|   |   +---:(source-host)
|   |   |   +---rw (src-address-or-name)
|   |   |   |   +---:(source-host-address)
|   |   |   |   |   +---rw spf-mac:source-host-address?
|   |   |   |   +---:(source-host-name)
|   |   |   |   |   +---rw spf-mac:source-host-name?
|   +---rw (dest-network)
|   |   +---:(dest-mac)
|   |   |   +---rw spf-mac:dest-address
|   |   |   +---rw spf-mac:dest-address-mask
|   |   +---:(dest-any)
|   |   |   +---rw spf-mac:dest-any?
|   |   +---:(dest-host)
|   |   |   +---rw (dest-address-or-name)
|   |   |   |   +---:(dest-host-address)
|   |   |   |   |   +---rw spf-mac:dest-host-address?
|   |   |   |   +---:(dest-host-name)
|   |   |   |   |   +---rw spf-mac:dest-host-name?
|   +---rw spf-mac:ethertype?
|   +---rw spf-mac:ethertype-mask?
|   +---rw spf-mac:cos?
|   +---rw spf-mac:time-range?
|   +---rw spf-mac:vlan?
|   +---rw spf-mac:enable-capture?
|   +---rw spf-mac:capture-session-id?
+---rw spf-mac:actions
|   +---rw spf-mac:action
|   +---rw spf-mac:log?
+---ro spf-mac:match?
+---rw spf-arp:arp-pfes
+---rw spf-arp:arp-pfe [name]
+---rw spf-arp:name
+---rw (remark-or-arp-pfe)?
+---:(remark)
|   +---rw spf-arp:remark?
+---:(arp-pfe)
+---rw spf-arp:filters
|   +---rw spf-arp:direction?
|   +---rw (source-address-host-group)

```

```

+---:(source-ip)
|   +---rw spf-arp:ip-source-address
|   +---rw spf-arp:ip-source-mask
+---:(ip-source-any)
|   +---rw spf-arp:ip-source-any?
+---:(source-host)
|   +---rw (ip-src-address-or-name)
|       +---:(ip-source-host-address)
|           +---rw spf-arp:ip-source-host-address?
|       +---:(ip-source-host-name)
|           +---rw spf-arp:ip-source-host-name?
+---:(source-group)
|   +---rw spf-arp:ip-source-group?
+---rw (dest-address-host-group)
|   +---:(dest-ip)
|       +---rw spf-arp:ip-dest-address
|       +---rw spf-arp:ip-dest-mask
+---:(ip-dest-any)
|   +---rw spf-arp:ip-dest-any?
+---:(dest-host)
|   +---rw (ip-dest-address-or-name)
|       +---:(ip-dest-host-address)
|           +---rw spf-arp:ip-dest-host-address?
|       +---:(ip-dest-host-name)
|           +---rw spf-arp:ip-dest-host-name?
+---:(dest-group)
|   +---rw spf-arp:ip-dest-group?
+---rw (source-network)
|   +---:(source-mac)
|       +---rw spf-arp:source-address
|       +---rw spf-arp:source-address-mask
+---:(source-any)
|   +---rw spf-arp:source-any?
+---:(source-host)
|   +---rw (src-address-or-name)
|       +---:(source-host-address)
|           +---rw spf-arp:source-host-address?
|       +---:(source-host-name)
|           +---rw spf-arp:source-host-name?
+---rw (dest-network)
|   +---:(dest-mac)
|       +---rw spf-arp:dest-address
|       +---rw spf-arp:dest-address-mask
+---:(dest-any)
|   +---rw spf-arp:dest-any?
+---:(dest-host)
|   +---rw (dest-address-or-name)
|       +---:(dest-host-address)

```

```

|         |         |         | +---rw spf-arp:dest-host-address?
|         |         |         | +---:(dest-host-name)
|         |         |         | +---rw spf-arp:dest-host-name?
|         |         |         | +---rw spf-arp:enable-capture?
|         |         |         | +---rw spf-arp:capture-session-id?
|         |         |         | +---rw spf-arp:actions
|         |         |         | +---rw spf-arp:action
|         |         |         | +---rw spf-arp:log?
|         |         |         | +---ro spf-arp:match?
+---rw port-groups
|   +---rw port-group [name]
|   |   +---rw name
|   |   +---rw port-group-entry [name]
|   |   |   +---rw name
|   |   |   +---rw (port-number-or-range)?
|   |   |   |   +---:(port-number-range)
|   |   |   |   |   +---rw port-lower
|   |   |   |   |   +---rw port-upper
|   |   |   |   +---:(port-number)
|   |   |   |   +---rw comparator
|   |   |   |   +---rw port
+---rw timerange-groups
|   +---rw timerange-group [name]
|   |   +---rw name
|   |   +---rw time-range [name]
|   |   |   +---rw name
|   |   |   +---rw remark?
|   |   |   +---rw (range-type)?
|   |   |   |   +---:(absolute)
|   |   |   |   |   +---rw absolute
|   |   |   |   |   |   +---rw start?
|   |   |   |   |   |   +---rw end?
|   |   |   |   +---:(periodic)
|   |   |   |   +---rw periodic
|   |   |   |   +---rw weekdays?
|   |   |   |   +---rw start?
|   |   |   |   +---rw end?
+---rw ip-address-groups
|   +---rw ip-address-group [name]
|   |   +---rw name
|   |   +---rw afi?
|   |   +---rw ip-address [name]
|   |   |   +---rw name
|   |   |   +---rw (ip-network-kind)
|   |   |   |   +---:(ip)
|   |   |   |   |   +---rw ip-address?
|   |   |   |   |   +---rw ip-mask
|   |   |   |   +---:(ip-any)

```

```

|   +---rw ip-any?
+---:(host)
    +---rw (address-or-name)
        +---:(ip-host-address)
            |   +---rw ip-host-address?
            +---:(ip-host-name)
                +---rw ip-host-name?

module: spf-ip
module: spf-mac
module: spf-arp

```

9. SPF Examples

9.1. Configuration Example

Requirement: Denies TELNET traffic from 14.3.6.234 bound for host 6.5.4.1 from leaving. Denies all TFTP traffic bound for TFTP servers. Permits all other IP traffic.

In order to achieve the requirement, an name access control list is needed. In the spf, we need three pfes. The spf and pfes can be described in CLI: as the following:

```

access-list ip ispf

deny tcp 14.3.6.234 0.0.0.0 host 6.5.4.1 eq 23
deny udp any any eq tftp
permit ip any any

```

Here is the example spf configuration xml:

```

<rpc message-id="101"
  xmlns:nc="urn:cisco:params:xml:ns:yang:spf:1.0"
  xmlns:spf-ip="urn:cisco:params:xml:ns:yang:spf-ip"
  // replace with IANA namespace when assigned
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <top xmlns="http://example.com/schema/1.2/config">

        <spfs>
          <spf >
            <name>sample-ip-spf</name>

```



```
<spf-type>ip-spf</spf-type>
<enable-match-counter>>false</enable-match-counter>
<spf-ip:afi>ipv4</spf-ip:afi>
<spf-ip:ipv4-pfes>

  <spf-ip:ipv4-pfe>
    <spf-ip:name>pfe10</spf-ip:name>
    <spf-ip:filters>
      <spf-ip:protocol>6</spf-ip:protocol>
      <spf-ip:ip-source-address>
        14.3.6.234
      </spf-ip:ip-source-address>
      <spf-ip:ip-source-mask>0.0.0.0</spf-ip:ip-source-mask>
      <spf-ip:ip-dest-host-address>
        6.5.4.1
      </spf-ip:ip-dest-host-address>
      <spf-ip:des-comparator>eq</spf-ip:des-comparator>
      <spf-ip:des-port>23</spf-ip:des-port>
    </spf-ip:filters>
    <spf-ip:actions>
      <spf-ip:action>deny</spf-ip:action>
    </spf-ip:actions>
  </spf-ip:ipv4-pfe>

  <spf-ip:ipv4-pfe>
    <spf-ip:name>pfe20</spf-ip:name>
    <spf-ip:filters>
      <spf-ip:protocol>17</spf-ip:protocol>
      <spf-ip:ip-source-any/>
      <spf-ip:ip-dest-any/>
      <spf-ip:des-comparator>eq</spf-ip:des-comparator>
      <spf-ip:des-port>69</spf-ip:des-port>
    </spf-ip:filters>
    <spf-ip:actions>
      <spf-ip:action>deny</spf-ip:action>
    </spf-ip:actions>
  </spf-ip:ipv4-pfe>

  <spf-ip:ipv4-pfe>
    <spf-ip:name>pfe30</spf-ip:name>
    <spf-ip:filters>
      <spf-ip:ip-source-any/>
      <spf-ip:ip-dest-any/>
    </spf-ip:filters>
    <spf-ip:actions>
      <spf-ip:action>permit</spf-ip:action>
    </spf-ip:actions>
  </spf-ip:ipv4-pfe>
```

```
        </spf-ip:ipv4-pfes>

        </spf>
    </spfs>

    </top>
</config>
</edit-config>
</rpc>
```

10. Stateless-PF YANG Module

This module imports type definitions from [RFC6021].

```
<CODE BEGINS> file "stateless-pf@2013-09-03.yang"
module stateless-pf {
    namespace "urn:cisco:params:xml:ns:yang:spf";
    // replace with IANA namespace when assigned
    prefix spf;

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

    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: netmod@ietf.org

        WG Chair: David Kessens
        david.kessens@nsn.com

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

        Editor: Lisa Huang
        yihuan@cisco.com

        Editor: Alexander Clemm
        alex@cisco.com
```

Editor: Andy Bierman
andy@yumaworks.com";

description

"This YANG module defines a component that describing the configuration of Stateless Packet Filters (SPF), also known as Access Control Lists (SPFs).

An SPF is an ordered set of rules and actions used to filter traffic. Each set of rules and actions is represented as an Packet Filter Entry (PFE), also known as Access Control Entries (PFE). Each PFE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

There are three types of SPF.

IP SPFs - IP SPFs are ordered sets of rules that can use to filter traffic based on IP information in the Layer 3 header of packets.

The device applies IP SPFs only to IP traffic. IP SPF can be IPv4 or IPv6.

MAC SPFs - MAC SPFs are used to filter traffic using the information in the Layer 2 header of each packet.

MAC SPFs are by default only applied to non-IP traffic; however, Layer 2 interfaces can be configured to apply MAC SPFs to all traffic.

ARP SPFs - The device applies ARP SPFs to IP traffic.

This module should be used with spf-ip, spf-arp, or spf-mac depends on what feature the device supports.

This YANG module also includes auxiliary definitions that are needed in conjunction with configuration of SPFs, such as reusable containers and references for ports and IP.

Terms and Acronyms

PFE (pfe): Packet Filter Entry

SPF (spf): Stateless Packet Filter

AFI (afi): Authority and Format Identifier (Address Field Identifier)

ARP (arp): Address Resolution Protocol

IP (ip): Internet Protocol

```
    IPv4 (ipv4): Internet Protocol Version 4

    IPv6 (ipv6): Internet Protocol Version 6

    MAC: Media Access Control

    TCP (tcp): Transmission Control Protocol

    TTL (ttl): Time to Live

    VLAN (vlan): Virtual Local Area Network
";

revision 2013-09-03 {
    description "Initial revision. ";
}

/* Features */

feature capture-session-id {
    if-feature packet-capture;
    description
        "The ability to configure SPF capture in order to
        selectively monitor traffic on an interface or VLAN.
        When the capture option for an SPF rule
        is enabled, packets that match this rule are
        either forwarded or dropped based on the specified permit
        or deny action and may also be copied to an alternate
        destination port for further analysis.
        An SPF rule with the capture option can be applied
        as follows:
            On a VLAN
            In the ingress direction on all interfaces
            In the egress direction on all Layer 3 interfaces
        The statistics data for the capture-session are capture
        in the device where the SPF rule applied to.";
}

feature host-by-name {
    description
        "The capability to reference a host by DNS name.";
}

feature ip-address-groups {
    description
        "The ability to define named groups for lists of
        ip addresses. ";
}
```

```
feature logging {
    description
        "The ability to log messages upon the matching of SPF's.";
}

feature match-counter {
    description
        "The ability to maintain global or local match statistics
        for each SPF rules.";
}

feature packet-capture {
    description "The ability to capture packets that
        match the filter.";
}

feature packet-length {
    description "The ability to filter packets by packet length";
}

feature port-groups {
    description
        "The ability to define named groups for lists of ports. ";
}

/* Identities */

identity spf-type {
    description "Base spf type for all SPF type identifiers.";
}

/* Types */

typedef spf-comparator {
    description "A data type used to express comparator string";
    type enumeration {
        enum "eq" {
            value 0;
            description "match only equal to any giving number.";
        }
        enum "gt" {
            value 1;
            description
                "match only greater than any giving number.";
        }
        enum "lt" {
```

```
        value 2;
        description
            "match only lower than any giving number.";
    }
    enum "neq" {
        value 3;
        description
            "match only not equal to any giving number";
    }
}

typedef spf-action {
    description "An enumeration data type to express spf
        action when match.";
    type enumeration {
        enum deny {
            description "Apply deny action to the traffic";
        }
        enum permit {
            description "Apply permit action to the traffic";
        }
    }
}

typedef spf-remark {
    type string {
        length "0..100";
    }
    description
        "A remark is a comment that can be
        associated with an PFE in order to make
        the access list easier for the network
        administrator to understand.
        It is retained to facilitate
        co-existence with CLI.";
}

typedef spf-type-ref {
    description
        "This type is used to refer to an Stateless Packet Filter
        (spf) type";
    type identityref {
        base "spf-type";
    }
}
```

```
typedef spf-ref {
    description "This type refers to an SPF.";
    type leafref {
        path "/spf:spfs/spf:spf/spf:name";
    }
}

typedef port-group-ref {
    description
        "This type is used to refer to a Portgroup object.";
    type leafref {
        path "/spfs/port-groups/port-group/name";
    }
}

typedef ip-address-group-ref {
    description
        "This type is used to refer to a time range object.";
    type leafref {
        path "/spfs/ip-address-groups/ip-address-group/name";
    }
}

typedef time-range-ref {
    description
        "This type is used to refer to a time range object.";
    type leafref {
        path "/spfs/timerange-groups/timerange-group/name";
    }
}

typedef weekdays {
    type bits {
        bit Sunday {
            position 0;
        }
        bit Monday {
            position 1;
        }
        bit Tuesday {
            position 2;
        }
        bit Wednesday {
            position 3;
        }
        bit Thursday {
```

```
        position 4;
    }
    bit Friday {
        position 5;
    }
    bit Saturday {
        position 6;
    }
}

typedef spf-name-string {
    type string {
        length "1 .. 64";
    }
}

/* Groupings */

grouping PFE-COMMON {
    description
        "A collection of nodes that should be added to
        every PFE list entry";

    container actions {
        leaf action {
            type spf:spf-action;
            mandatory true;
            description "Permit/deny action.";
        }

        leaf log {
            if-feature spf:logging;
            type empty;
            description
                "Causes an informational logging message about the
                packet that matches the entry to be sent to the
                console.";
        }
    }

    leaf match {
        if-feature spf:match-counter;
        config false;
        type yang:counter64;
        description
            "The total packet that have matched for the
            particular PFE";
    }
}
```



```
    }  
  }  
  
  grouping FILTER-COMMON {  
    description  
      "A collection of nodes that should be added to  
      every 'filters' container within each  
      PFE list entry";  
  
    leaf enable-capture {  
      if-feature spf:packet-capture;  
      type boolean;  
      description  
        "Enable packet capture on this filter  
        for this session.";  
    }  
  
    leaf capture-session-id {  
      if-feature spf:capture-session-id;  
      when "../enable-capture = 'true'";  
      type uint32 {  
        range "1..48";  
      }  
      description  
        "Enable packet capture on this filter  
        for this session id.";  
    }  
  }  
}  
  
/* Data Nodes */  
  
container spfs {  
  description  
    "This is the top container that contains a list of  
    named SPF and reusable spf object groups.";  
  list spf {  
    key name;  
    leaf name {  
      description "spf/access group name.";  
      type spf-name-string;  
    }  
  
    leaf spf-type {  
      type spf-type-ref;  
      description "Type of SPF";  
      mandatory true;  
    }  
    leaf enable-capture-global {
```

```
        if-feature packet-capture;
        type boolean;
        description "Enable packet capture on this filter
            for this session. Session ID range is 1 to 48";
        default "false";
    }
    leaf capture-session-id-global {
        if-feature capture-session-id;
        when "../enable-capture-global = 'true'";
        type uint32 {
            range "1..48";
        }
        description "Enable packet capture on this filter
            for this session. Session ID range is 1 to 48";
    }
    choice enable-match-counter-choices {
        if-feature match-counter;
        case match {
            leaf enable-match-counter {
                type boolean;
                description
                    "Enable to collect statistics for the SPF";
                default false;
            }
        }
        case per-entry-match {
            leaf enable-per-entry-match-counter {
                type boolean;
                description "Enable to collect match
                    statistics for each SPF entry(Stateless PFE).";
                default false;
            }
        }
    }

    leaf match {
        if-feature match-counter;
        config false;
        type yang:counter64;
        description
            "The total packet that have matched for the
            particular access list";
    }
}

container port-groups {
    if-feature port-groups;
```

```
list port-group {
  key "name";
  leaf name {
    type spf-name-string;
  }
  list port-group-entry {
    key "name";
    ordered-by user;
    leaf name {
      type spf-name-string;
    }
    //unique "comparator port-number
    //port-lower port-upper";

    choice port-number-or-range {
      case port-number-range {
        description
          "Port group includes all ports between
          port-lowerand port-upper (including those)";
        leaf port-lower {
          type inet:port-number;
          description "Lower Port number.";
          mandatory true;
        }
        leaf port-upper {
          type inet:port-number;
          description "Upper Port number.";
          mandatory true;
          must "../port-lower <= ../port-upper";
        }
      }
    }
    case port-number {
      description
        "Port group includes all ports that are greater
        than, greater or equal, less than, less or
        equal, or not equal the port, per the
        indicated comparator.
        It is possible for the port group to be empty
        (for example, in case a port group that
        is less than the minimum port number is
        specified).";
      leaf comparator {
        type spf-comparator;
        mandatory true;
      }
      leaf port {
        type inet:port-number;
        description "Port number.";
      }
    }
  }
}
```

```
        mandatory true;
    }
}
} // choice port-number-or-range
} // list port-group-entry
} // list port-group
} // container port-groups

container timerange-groups {
    description "Define time range entries to restrict
        the access. The time range is identified by a name
        and then referenced by a function, so that those
        time restrictions are imposed on the function itself.";
    list timerange-group {
        key "name";
        leaf name {
            type spf-name-string;
        }
        list time-range {
            key "name";
            ordered-by user;
            leaf name {
                type spf-name-string;
            }

            leaf remark {
                type spf-remark;
            }

            choice range-type {
                // absolute or periodic time range
                container absolute {
                    description
                        "Absolute time and date that
                        the associated function starts
                        going into effect.";
                    leaf start {
                        type yang:date-and-time;
                        description
                            "Absolute start time and date";
                    }
                    leaf end {
                        type yang:date-and-time;
                        description "Absolute end time and date";
                    }
                }
            }
        }
        container periodic {
            description
```

```
        "To specify a periodic time and date.";
    leaf weekdays {
        type weekdays;
    }
    leaf start {
        type yang:timestamp;
        description "Start time";
    }
    leaf end {
        type yang:timestamp;
        description "End time";
    }
}
} // choice range-type
} // list time-range
} // list timerange-group
} // container timerange-groups

container ip-address-groups {
    if-feature ip-address-groups;
    description
        "This contains a list of named ip address group. Each
        group defines a range of address and mask pair.";
    list ip-address-group {
        key "name";
        leaf name {
            type spf-name-string;
        }
        leaf afi {
            default "ipv4";
            type inet:ip-version;
            description "Address Field Identifier (AFI).";
        }
    }
    list ip-address {
        key "name";
        ordered-by user;
        leaf name {
            type spf-name-string;
        }
        //unique "ip-address ip-mask";
        //unique "ip-host-address";

        grouping IP-HOST {
            description
                "Choice within a case not allowed so need
                this grouping.";
            choice address-or-name {
                mandatory true;
            }
        }
    }
}
```

```

        leaf ip-host-address {
            type inet:ip-address;
        }
        leaf ip-host-name {
            if-feature spf:host-by-name;
            type inet:domain-name;
        }
    }
}

choice ip-network-kind {
    mandatory true;

    case ip {
        leaf ip-address {
            type inet:ip-address;
        }
        leaf ip-mask {
            type inet:ip-prefix;
            mandatory true;
        }
    }
    leaf ip-any {
        type empty;
        description "To express Any network or address.
            Use the any keyword as an abbreviation
            for an address and a mask of 0.0.0.0
            255.255.255.255. For example:
            0.0.0.0/255.255.255.255 means 'any'";
    }
    case host {
        description
            "Use the host address combination as an
            abbreviation for an address and wildcard
            of address 0.0.0.0";

        uses IP-HOST;
    }
    // case group not allowed here!
}

    } // list ip-address
    } // list ip-address-group
    } // container ip-address-groups
} // container spfs
}

```

<CODE ENDS>

11. SPF-IP YANG Module

This module imports type definitions from [RFC6021] and common-types yang defined with stateless-pf model.

```
<CODE BEGINS> file "spf-ip@2013-09-03.yang"
module spf-ip {
    namespace "urn:cisco:params:xml:ns:yang:spf-ip";
    // replace with IANA namespace when assigned
    prefix spf-ip;

    import stateless-pf {
        prefix spf;
    }
    import ietf-inet-types {
        prefix "inet";
    }
    import common-types {
        prefix "c-types";
    }

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

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

        WG Chair: David Kessens
        david.kessens@nsn.com

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

        Editor: Lisa Huang
        yihuan@cisco.com

        Editor: Alexander Clemm
        alex@cisco.com

        Editor: Andy Bierman
        andy@yumaworks.com";

    description
        "This YANG module augments the 'stateless-pf' module with configuratio
```

n

and operational data for IPv4 and IPv6 stateless packet filter.

An Stateless Packet Filter (SPF), also know as an Access Control List (SPF), is an ordered set of rules and actions used to filter traffic. Each set of rules and actions is represented as a Packet Filter Entry (PFE), also know as an Access Control Entries (PFE). Each PFE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

IP SPFs are ordered sets of rules that can use to filter traffic based on IP information in the Layer 3 header of packets.

The device applies IP SPFs only to IP traffic. IP SPF can be IPv4 or IPv6.

Terms and Acronyms

PFE (pfe): Packet Filter Entry

SPF (spf): Stateless Packet Filter

AFI (afi): Authority and Format Identifier (Address Field Identifier)

DSCP (dscp): Differentiated Services Code Point

ICMP (icmp): Internet Control Message Protocol

IGMP (igmp): Internet Group Management Protocol

IP (ip): Internet Protocol

IPv4 (ipv4):Internet Protocol Version 4

IPv6 (ipv6): Internet Protocol Version 6

QoS: Quality of Service

TCP (tcp): Transmission Control Protocol

ToS (tos): Type of Service

TTL (ttl): Time to Live

UDP (udp): User Datagram Protocol


```
VLAN (vlan): Virtual Local Area Network

VRF(vrf) : Virtual Routing and Forwarding
";

revision 2013-09-03 {
    description "Initial revision. ";
}

/* Features */

feature time-to-live {
    description "The ability to filter packets based on their
        time-to-live (TTL) value (0 to 255)";
    reference "SPF Support for Filtering on TTL Value";
}

feature flow-label {
    description
        "The ability to filter packets based on flow lable.
        The 20-bit Flow Label field in the IPv6 header
        is used by a source to label packets
        of a flow. This is an IPv6 PFEs option.";
    reference "RFC 3697 IPv6 Flow Label Specification";
}

/* Identities */

identity ip-spf {
    base "spf:spf-type";
    description "layer 3 SPF type";
}

/* Groupings */

grouping IP-SOURCE-NETWORK {
    description "Reusable IP address and mask pair.";

    grouping IP-SOURCE-HOST {
        description
            "Choice within a case not allowed so need
            this grouping.";
        choice ip-src-address-or-name {
            mandatory true;
            leaf ip-source-host-address {
                type inet:ip-address;
            }
        }
    }
}
```

```

        leaf ip-source-host-name {
            if-feature spf:host-by-name;
            type inet:domain-name;
        }
    }
}

choice source-address-host-group {
    mandatory true;
    case source-ip {
        description "Used with address and mask couple
            to express network.";

        leaf ip-source-address {
            type inet:ip-address;
            mandatory true;
        }
        leaf ip-source-mask {
            type inet:ip-address;
            mandatory true;
        }
    }
    leaf ip-source-any {
        type empty;
        description "To express Any network or address.
            Use the any keyword as an abbreviation
            for an address and a mask of 0.0.0.0
            255.255.255.255. For example:
            0.0.0.0/255.255.255.255 means 'any'";
    }
    case source-host {
        description "Used with host address to express a
            single host
            Use the host address(or name)
            combination is the same as an address
            and mask of address 0.0.0.0.
            For example: '10.1.1.2/0.0.0.0' is the same
            as 'host 10.1.1.2'";
        uses IP-SOURCE-HOST;
    }
    case source-group {
        if-feature spf:ip-address-groups;
        leaf ip-source-group {
            type spf:ip-address-group-ref;
        }
    }
}
}

```

```
grouping IP-DESTINATION-NETWORK {
  description
    "Reusable IP address and mask pair for destination.";

  grouping IP-DESTINATION-HOST {
    description
      "Choice within a case not allowed so need
      this grouping.";
    choice ip-dest-address-or-name {
      mandatory true;
      leaf ip-dest-host-address {
        type inet:ip-address;
      }
      leaf ip-dest-host-name {
        if-feature spf:host-by-name;
        type inet:domain-name;
      }
    }
  }

  choice dest-address-host-group {
    mandatory true;
    case dest-ip {
      description "Used with address and mask couple
        to express network.";
      leaf ip-dest-address {
        type inet:ip-address;
        mandatory true;
      }
      leaf ip-dest-mask {
        type inet:ip-address;
        mandatory true;
      }
    }
    leaf ip-dest-any {
      type empty;
      description "To express Any network or address.
        Use the any keyword as an abbreviation
        for an address and a mask of 0.0.0.0
        255.255.255.255. For example:
        0.0.0.0/255.255.255.255 means 'any'";
    }
    case dest-host {
      description "Used with host address to express a
        single host
        Use the host address(or name)
        combination is the same as an address
        and mask of address 0.0.0.0."
    }
  }
}
```

For example: '10.1.1.2/0.0.0.0' is the same
as 'host 10.1.1.2'";

```

        uses IP-DESTINATION-HOST;
    }
    case dest-group {
        if-feature spf:ip-address-groups;
        description "Use the group keyword and group name
            to refer to a pre-defined address object group
            which is a list of address and mask.";

        leaf ip-dest-group {
            type spf:ip-address-group-ref;
        }
    }
}

grouping DSCP-OR-TOS {
    choice dscp-or-tos {
        leaf dscp {
            type inet:dscp;
            description
                "Match packets with given dscp value";
        }

        case tos {
            leaf tos {
                type c-types:tos;
                description
                    "Match packets with given TOS value";
            }
            leaf precedence {
                when "boolean(..../tos)" ;
                type c-types:precedence;
                description
                    "Match packets with given precedence value";
            }
        }
    }
}

grouping IP-PFE-FILTERS {
    leaf protocol {
        type c-types:ip-protocol;
        description "IP protocol number.";
    }
}

```

```
uses spf:FILTER-COMMON;

leaf fragments {
  type empty;
  description "Check non-initial fragments";
}

leaf time-range {
  type spf:time-range-ref;
  description
    "Refer a time range object by
     name (Max Size 64).";
}

choice src-ports {
  when "protocol = '6' or protocol = '17' or " +
        "protocol = '132'";

  description
    "Apply only when the protocol is TCP,
     UDP or SCTP.";

  case port-number-range {
    description
      "Port group includes all ports between port-lower
       and port-upper (including those)";
    leaf src-port-lower {
      type inet:port-number;
      description "Lower Port number.";
      mandatory true;
    }
    leaf src-port-upper {
      type inet:port-number;
      description "Upper Port number.";
      mandatory true;
      must "../src-port-lower <= ../src-port-upper";
    }
  }
  case port-number {
    description
      "Port group includes all ports that are greater
       than, greater or equal, less than, less or equal,
       or not equal the port, per the indicated
       comparator. It is possible for the port group
       to be empty (for example, in case a port group
       that is less than the minimum port number is
       specified).";
    leaf src-comparator {
```

```
        type spf:spf-comparator;
        mandatory true;
    }
    leaf src-port {
        type inet:port-number;
        description "Port number.";
        mandatory true;
    }
}
case port-group-ref {
    if-feature spf:port-groups;
    leaf src-port-group-name {
        type spf:port-group-ref;
        mandatory true;
        description
            "Reference a port group by the Port
            Group name.";
    }
}
} // choice src-ports

choice dest-ports {
    when "protocol = '6' or protocol = '17' or " +
        "protocol = '132'";
    description
        "Apply only when the protocol is TCP,
        UDP or SCTP.";

    case port-number-range {
        description "Port group includes all ports between
            port-lower and port-upper (including those)";
        leaf des-port-lower {
            type inet:port-number;
            description "Lower Port number.";
            mandatory true;
        }
        leaf des-port-upper {
            type inet:port-number;
            description "Upper Port number.";
            mandatory true;
            must "../des-port-lower <= ../des-port-upper";
        }
    }
}
case port-number {
    description "Port group includes all ports that
        are greater than, greater or equal, less than,
        less or equal, or not equal the port, per the
        indicated comparator. It is possible for the
```

```
        port group to be empty (for example, in case a
        port group that is less than the minimum port
        number is specified).";
    leaf des-comparator {
        type spf:spf-comparator;
        mandatory true;
    }
    leaf des-port {
        type inet:port-number;
        description "Port number.";
        mandatory true;
    }
}
case port-group-ref {
    if-feature spf:port-groups;
    leaf des-port-group-name {
        type spf:port-group-ref;
        mandatory true;
        description
            "Reference a port group by the Port Group name.";
    }
}
} // choice dest-ports

leaf icmp-type {
    when "../protocol = '1'";
    type c-types:icmp-type;
    description
        "ICMP message type number.
        Apply only when the protocol is icmp";
}

leaf icmp-code {
    when "boolean(..icmp-type) ";
    type c-types:icmp-code;
    description
        "ICMP subtype for a given icmp type.";
}

choice packet-length-or-range {
    if-feature spf:packet-length;
    case length {
        leaf packet-length-comparator {
            type spf:spf-comparator;
            description
                "Operand that compare the packet
                length. Operands are lt (less than),
                gt (greater than), eq (equal), and neq
```

```
        (not equal).";
        mandatory true;
    }
    leaf packet-length {
        type uint32 {
            range "20..9210";
        }
        description
            "Packet length value for
            operation gt, eq, etc, other
            than range";
        //TODO need to find out why package is
        // less than 9210
        mandatory true;
    }
}
case range {
    description
        "Packet operator 'range' takes
        both lower and upper value.";

    leaf packet-length-upper {
        type uint32 {
            range "20..9210";
        }
        mandatory true;
        description "Upper Packet length";
    }

    leaf packet-length-lower {
        type uint32 {
            range "20..9210";
        }
        must "number(..../packet-length-lower) <= " +
            "number(..../packet-length-upper)";
        mandatory true;
        description "Lower packet length";
    }
}

}

leaf tcp-flag-value {
    type c-types:tcp-flag-type ;
    description "TCP flag bits that needs to be checked";
}

leaf tcp-flag-mask {
    when "boolean(..../tcp-flag-value)" ;
}
```



```
    type c-types:tcp-flag-type ;
    description "TCP flag bit that needs to be checked";
}

leaf tcp-flag-operation {
    when "boolean(..tcp-flag-value)" ;
    description
        "TCP flag Match option.
        A match occurs if the TCP
        datagram has certain TCP flags
        set or not set. You use the
        match-any keyword to allow a match
        to occur if any of the specified
        TCP flags are present, or you can
        use the match-all keyword to allow
        a match to occur only if all of
        the specified TCP flags are
        present. You must follow the
        match-any and match-all keywords
        with the + or - keyword and the
        flag-name argument to match on
        one or more TCP flags. ";
    default match-any;
    type enumeration {
        enum match-any {
            description "match any";
        }
        enum match-all {
            description "match all";
        }
    }
}

choice ttl-value-or-range {
    if-feature time-to-live;
    case value {
        leaf ttl-comparator {
            type spf:spf-comparator;

            description
                "Compares the TTL value in the packet
                to the TTL value specified in this
                PFE statement. Operands are lt (less
                than), gt (greater than), and eq
                (equal), neq (not equal).";
        }
        leaf ttl-value {
            type c-types:time-to-live;
        }
    }
}
```

```

    }
  }
  case range {
    leaf ttl-value-lower {
      type c-types:time-to-live;
      description "Lower ttl number.";
    }
    leaf ttl-value--upper {
      type c-types:time-to-live;
      description "Upper ttl number.";
    }
  }
}

/* Data Nodes */

augment "/spf:spfs/spf:spf" {
  when "spf:spf-type = 'ip-spf'";

  leaf afi {
    type inet:ip-version ;
    default "ipv4";
  }

  container ipv6-pfes {
    when "../afi = 'ipv6' " ;

    description
      " The ip-pfes container contains a list of ip-pfe.
      Each ip-pfe is made of a unique ID, an optional
      remark (comment), and a filter. The filter
      requires a mandatory action (permit/deny) and one or
      more options such as source-address with mask,ttl etc";

    list ipv6-pfe {
      key "name";
      ordered-by user;
      description "Layer 3 Packet Filter Entry (PFE)";

      leaf name {
        type spf:spf-name-string;
        description "Unique PFE identifier.";
      }

      choice remark-or-ipv6-case {
        leaf remark {

```

```
    type spf:spf-remark;
    // mandatory true;
  }
  case ipv6-pfe {
    container filters {

      uses IP-SOURCE-NETWORK;
      uses IP-DESTINATION-NETWORK;
      uses IP-PFE-FILTERS;
      uses DSCP-OR-TOS;

      leaf igmp-type {
        when "../protocol = '2' ";
        type c-types:igmp-code;
        description
          "IGMP message type (0 to 15) for
           filtering IGMP packets. Apply only
           when the protocol is igmp in ipv4";
      }

      leaf flow-label {
        if-feature flow-label;
        when "../protocol = '17' ";
        type uint64 {
          range "0..1048575";
        }
        description
          "Flow label value. Apply only when
           the protocol is UDP in ipv6.";
        reference
          "RFC3697 IPv6 Flow Label Specification";
      }
    } // container filters

    uses spf:PFE-COMMON;
  } // case ipv6-pfe
} // choice remark-or-ipv6-pfe
} // list ipv6-pfe
} // container ipv6-pfes

container ipv4-pfes {
  when "../afi = 'ipv4' " ;

  description
    "The ip-pfes container contains a list of ip-pfe.
    Each ip-pfe is made of a unique ID, an optional
    remark (comment), and a filter. The filter requires a
    mandatory action (permit/deny) and one or more options
```

```
such as source-address with mask,ttl etc";

list ipv4-pfe {
  key "name";
  ordered-by user;
  description "Layer 3 Packet Filter Entry (PFE)";

  leaf name {
    type spf:spf-name-string;
    description "Unique PFE identifier";
  }

  choice remark-or-ipv4-pfe {
    leaf remark {
      type spf:spf-remark;
      // mandatory true;
    }
    case ipv4-pfe {
      container filters {
        uses IP-SOURCE-NETWORK;
        uses IP-DESTINATION-NETWORK;
        uses IP-PFE-FILTERS;
        uses DSCP-OR-TOS;
      }
      uses spf:PFE-COMMON;
    } // case ipv4-pfe
  } // choice remark-or-ipv4-pfe
} // list ipv4-pfe
} // container ipv4-pfes

leaf global-fragments {
  default "not-set";
  type enumeration {
    enum not-set;
    enum permit-all {
      description "Allow all fragments";
    }
    enum deny-all {
      description "Drop all fragments";
    }
  }
}
description
  "Optimizes fragment handling for noninitial fragments.
  When this leaf is set to 'permit-all', noninitial
  fragments will be permitted unless explicitly denied.
  When this leaf is set to 'deny-all', noninitial
  fragments will be denied unless explicitly
  permitted. ";
```

```
    }  
  }  
}
```

<CODE ENDS>

12. SPF-MAC Configuration YANG Module

This module imports type definitions from common-types YANG defined in this model.

<CODE BEGINS> file "spf-mac@2013-09-03.yang"

```
module spf-mac {  
  namespace "urn:cisco:params:xml:ns:yang:spf-mac";  
  // replace with IANA namespace when assigned  
  prefix spf-mac;  
  
  import stateless-pf { prefix spf; }  
  
  import common-types {  
    prefix "c-types";  
  }  
  
  import ietf-inet-types {  
    prefix "inet";  
  }  
  
  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: netmod@ietf.org  
  
    WG Chair: David Kessens  
    david.kessens@nsn.com  
  
    WG Chair: Juergen Schoenwaelder  
    j.schoenwaelder@jacobs-university.de
```

Editor: Lisa Huang
yihuan@cisco.com

Editor: Alexander Clemm
alex@cisco.com

Editor: Andy Bierman
andy@yumaworks.com";

description

"This YANG module augments the 'stateless-pf' module with configuration and operational data for MAC stateless packet filter.

An Stateless Packet Filter (SPF), also know as an Access Control List (SPF), is an ordered set of rules and actions used to filter traffic. Each set of rules and actions is represented as a Packet Filter Entry (PFE), also know as an Access Control Entries (PFE). Each PFE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

MAC SPFs - MAC SPFs are used to filter traffic using the information in the Layer 2 header of each packet. MAC SPFs are by default only applied to non-IP traffic; however, Layer 2 interfaces can be configured to apply MAC SPFs to all traffic.

Terms and Acronyms

PFE (pfe): Packet Filter Entry

SPF (spf): Stateless Packet Filter

AFI (afi): Authority and Format Identifier (Address Field Identifier)

CoS (cos): Class of Service

MAC: Media Access Control

TTL (ttl): Time to Live

VLAN (vlan): Virtual Local Area Network

VRF(vrf) : Virtual Routing and Forwarding
";

```
revision 2013-09-03 {
    description "Initial revision. ";
}

/* Features */

feature ethertype-mask {
    description
        "The ability to filter packets based on ether-type mask
        in hex 0x0-0xFFFF.";
}

/* Identities */

identity mac-spf {
    base spf:spf-type;
    description "layer 2 SPF type";
}

/* Groupings */

grouping MAC-SOURCE-NETWORK {
    description "MAC address and mask pair for source.";

    grouping MAC-SOURCE-HOST {
        description
            "Choice within a case not allowed so need
            this grouping.";
        choice src-address-or-name {
            mandatory true;
            leaf source-host-address {
                type inet:ip-address;
                description
                    "Use the host address combination as an
                    abbreviation for an address and wildcard
                    of address 0.0.0.0";
            }
            leaf source-host-name {
                if-feature spf:host-by-name;
                type inet:domain-name;
            }
        }
    }

    choice source-network {
        mandatory true;
        case source-mac {
            description
```

```
        "Used with address and mask couple to
        express network.";
    leaf source-address {
        type yang:mac-address;
        mandatory true;
        description "A source MAC address.";
    }
    leaf source-address-mask {
        type yang:mac-address;
        mandatory true;
        description "A source MAC address mask.";
    }
}
leaf source-any {
    type empty;
    description "To express Any network or address";
}
case source-host {
    description
        "Use the host address combination as an
        abbreviation for an address and wildcard
        of address 0.0.0.0";
    uses MAC-SOURCE-HOST;
}
}

grouping MAC-DESTINATION-NETWORK {
    description
        "MAC address and mask pair for destination.";

    grouping MAC-DESTINATION-HOST {
        description
            "Choice within a case not allowed so need
            this grouping.";
        choice dest-address-or-name {
            mandatory true;
            leaf dest-host-address {
                type inet:ip-address;
                description
                    "Use the host address combination as an
                    abbreviation for an address and wildcard
                    of address 0.0.0.0";
            }
            leaf dest-host-name {
                if-feature spf:host-by-name;
                type inet:domain-name;
            }
        }
    }
}
```



```

    }
  }

  choice dest-network {
    mandatory true;
    case dest-mac {
      description
        "Used with address and mask couple to
        express network.";
      leaf dest-address {
        type yang:mac-address;
        mandatory true;
        description "A source MAC address.";
      }
      leaf dest-address-mask {
        type yang:mac-address;
        mandatory true;
        description "A source MAC address mask.";
      }
    }
    leaf dest-any {
      type empty;
      description "To express Any network or address";
    }
    case dest-host {
      description
        "Use the host address combination as an
        abbreviation for an address and wildcard
        of address 0.0.0.0";
      uses MAC-DESTINATION-HOST;
    }
  }
}

/* Layer 2 SPF */

augment "/spf:spf/spf:spf" {
  when "spf:spf-type = 'mac-spf'";
  description
    "Layer 2 Packet Filter Entry (PFE). The mac-pfes
    container contains a list of mac-pfe. Each mac-pfe is
    comprised of a name, an optional remark
    and a rule.
    A rule is referred to as 'packet-filter', although it
    contains both a filter and an action.
    The packet-filter requires a mandatory action (permit/deny)
    and one or more options such as source-address with mask,
    ethertype, vlan etc.";

```

```
container mac-pfes {
  list mac-pfe {
    key name;
    ordered-by user;

    leaf name {
      type spf:spf-name-string;
      description "Unique PFE identifier";
    }

    choice remark-or-mac-pfe {
      leaf remark {
        type spf:spf-remark;
        // mandatory true;
      }
      case mac-pfe {
        container filters {
          uses MAC-SOURCE-NETWORK;
          uses MAC-DESTINATION-NETWORK;

          leaf ethertype {
            type c-types:ether-type;
            description "Ether-Type (also known as
              protocol) in hex 0x0-0xffff";
          }

          leaf ethertype-mask {
            if-feature ethertype-mask;
            when "boolean(..ethertype)";
            type c-types:ether-type;
            default "0x0000";
            description
              "Ether-type mask in hex 0x0-0xFFFF.
              0x0 is exactly match of the Ethertype..";
          }

          leaf cos {
            type c-types:cos;
            description "CoS value <0-7>";
          }

          leaf time-range {
            type spf:time-range-ref;
            description
              "Enable packet capture on this
              filter for a specify time range
              by name.";
          }
        }
      }
    }
  }
}
```

```
        leaf vlan {
            type c-types:vlan-identifier;
            description "VLAN number";
        }

        uses spf:FILTER-COMMON;

    } // container filters

    uses spf:PFE-COMMON;

    } // case mac-pfe
    } // choice remark-or-pfe
    } // list mac-pfe
    } // container mac-pfes
} // augment

}
```

<CODE ENDS>

13. SPF-ARP Configuration YANG Module

<CODE BEGINS> file "spf-arp@2013-09-03.yang"

```
module spf-arp {
    namespace "urn:cisco:params:xml:ns:yang:spf-arp";
    // replace with IANA namespace when assigned
    prefix spf-arp;

    import stateless-pf { prefix spf; }
    import spf-ip { prefix spf-ip; }
    import spf-mac { prefix spf-mac; }

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

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

        WG Chair: David Kessens
        david.kessens@nsn.com

        WG Chair: Juergen Schoenwaelder
```

j.schoenwaelder@jacobs-university.de

Editor: Lisa Huang
yihuan@cisco.com

Editor: Alexander Clemm
alex@cisco.com

Editor: Andy Bierman
andy@yumaworks.com";

description

"This YANG module augments the 'stateless-pf' module with configuration and operational data for ARP stateless packet filter.

An Stateless Packet Filter (SPF), also know as an Access Control List (SPF), is an ordered set of rules and actions used to filter traffic. Each set of rules and actions is represented as a Packet Filter Entry (PFE), also know as an Access Control Entries (PFE). Each PFE is evaluated sequentially. When the rule matches then action for that rule is applied to the packet.

ARP SPFs - The device applies ARP SPFs to IP traffic.

Terms and Acronyms

PFE (pfe): Packet Filter Entry

SPF (spf): Stateless Packet Filter

ARP (arp): Address Resolution Protocol

IP (ip): Internet Protocol

MAC: Media Access Control

VLAN (vlan): Virtual Local Area Network
";

```
revision 2013-09-03 {  
    description "Initial revision. ";  
}
```

```
/* Identities */
```

```
identity arp-spf {  
    base "spf:spf-type";
```

```
    description "ARP SPF type";
  }

  /* Data Nodes */

  augment "/spf:spfs/spf:spf" {
    when "spf:spf-type = 'arp-spf'";

    description "ARP Packet Filter Entry (PFE).";
    container arp-pfes {
      list arp-pfe {
        key "name";
        ordered-by user;

        leaf name {
          type spf:spf-name-string;
        }

        choice remark-or-arp-pfe {
          leaf remark {
            type spf:spf-remark;
            // mandatory true;
          }
          case arp-pfe {
            container filters {
              leaf direction {
                default "bi-direction";
                type enumeration {
                  enum bi-direction;
                  enum request;
                  enum response;
                }
                description "ARP request/response.";
              }
            }

            uses spf-ip:IP-SOURCE-NETWORK;
            uses spf-ip:IP-DESTINATION-NETWORK {
              when "../direction = 'response'";
            }

            uses spf-mac:MAC-SOURCE-NETWORK;
            uses spf-mac:MAC-DESTINATION-NETWORK {
              when "../direction = 'response'";
            }

            uses spf:FILTER-COMMON;
          } // container filters
        }
      }
    }
  }
```

```
        uses spf:PFE-COMMON;

        } // case arp-pfe
      } // choice remark-or-arp-pfe
    } // list arp-pfe
  } // container arp-pfes
} // augment

}

<CODE ENDS>
```

14. COMMON-TYPES YANG Module

```
<CODE BEGINS> file "common-types@2012-10-12.yang"

module common-types {
  namespace "urn:cisco:params:xml:ns:yang:common-types";
  // replace with IANA namespace when assigned
  prefix c-types;

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

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

    WG Chair: David Kessens
    david.kessens@nsn.com

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

    Editor: Lisa Huang
    yihuan@cisco.com

    Editor: Alexander Clemm
    alex@cisco.com

    Editor: Andy Bierman
    andy@yumaworks.com";

  description
    "This module contains a collection of generally useful
    YANG types could be referred from multiple speciality
```

components.

Terms and Acronyms

CoS (cos): Class of Service

ICMP (icmp): Internet Control Message Protocol

IGMP (igmp): Internet Group Management Protocol

IP (ip): Internet Protocol

IPv4 (ipv4): Internet Protocol Version 4

IPv6 (ipv6): Internet Protocol Version 6

TCP (tcp): Transmission Control Protocol

ToS (tos): Type of Service

TTL (ttl): Time to Live

UDP (udp): User Datagram Protocol

VLAN (vlan): Virtual Local Area Network
";

```
revision 2012-10-12 {  
    description "Initial revision. ";  
}
```

```
/* Typedefs */
```

```
typedef cos {  
    type uint8 {  
        range "0..7";  
    }  
    description  
        "Class of Service.  
        An integer that is in the range of the layer 2 CoS values.  
        This corresponds to the 802.1p and ISL CoS values."  
    reference "IEEE 802.1p";  
}
```

```
typedef tos {  
    type uint8 {  
        range "0..15";  
    }  
    description
```

"tos stands for Type of service .
 The tos field are five bits in the IPv4 header.
 It could specify a datagrams priority and
 request a route for low-delay, high-throughput,
 or highly-reliable service.

Based on these TOS values, a packet would be placed in
 an prioritized outgoing queue, or take a route with
 appropriate latency, throughput, or reliability.
 The following are TOS field values (expressed as
 binary numbers):

1000	--	minimize delay
0100	--	maximize throughput
0010	--	maximize reliability
0001	--	minimize monetary cost
0000	--	normal service

.";

reference

"RFC 791 Internet Protocol
 Protocol Specification
 RFC 1122 Requirements for Internet Hosts --
 Communication Layers
 RFC 1349 Type of Service in the Internet Protocol
 Suite
 RFC 2474 Definition of the Differentiated Services
 Field (DS Field)
 in the IPv4 and IPv6 Headers
 RFC 3168 The Addition of Explicit Congestion
 Notification (ECN) to IP
 ";

}

```
typedef precedence {
    type uint8 {
        range "0..7";
    }
}
```

description

"Indicates the IP precedence.
 Precedence is three bits in IP header.

Value	Description
000 (0)	Routine or Best Effort
001 (1)	Priority
010 (2)	Immediate


```
011 (3)      Flash - mainly used for Voice Signaling
              or for Video.
100 (4)      Flash Override
101 (5)      Critical -mainly used for Voice RTP.
110 (6)      Internet
111 (7)      Network";
```

reference

```
"RFC 791 Internet Protocol Chapter 3.1
Protocol Specification";
```

```
}
```

```
typedef tcp-flag-type {
  type bits {
    bit fin {
      position 0;
      description "No more data from sender";
    }
    bit syn {
      position 1;
      description "Synchronize sequence numbers";
    }
    bit rst {
      position 2;
      description "Reset the connection";
    }
    bit psh {
      position 3;
      description "Push Function";
    }
    bit ack {
      position 4;
      description "Acknowledgment field significant";
    }
    bit urg {
      position 5;
      description "Urgent Pointer field significant";
    }
  }
  description "TCP flag type";
  reference "RFC 793 TRANSMISSION CONTROL PROTOCOL";
}
```

```
typedef ether-type {
  type string {
    pattern '0x[0-9a-fA-F]{4}';
  }
  description
```

"ether-type is 0x0-0xffff. The protocol number is a four-byte hexadecimal number prefixed with 0x. Valid protocol numbers are from 0x0 to 0xffff.

This list shows the EtherType values and their corresponding protocol keywords:

0x0600 xns-idp Xerox XNS IDP

0x0BAD vines-ip Banyan VINES IP

0x0baf vines-echo Banyan VINES Echo

0x6000 etype-6000 DEC unassigned, experimental

0x6001 mop-dump DEC Maintenance Operation Protocol
(MOP) Dump/Load Assistance

0x6002 mop-console DEC MOP Remote Console

0x6003 decnet-iv DEC DECnet Phase IV Route

0x6004 lat DEC Local Area Transport (LAT)

0x6005 diagnostic DEC DECnet Diagnostics

0x6007 lavc-sca DEC Local-Area VAX Cluster (LAVC), SCA

0x6008 amber DEC AMBER

0x6009 mumps DEC MUMPS

0x0800 ip Malformed, invalid, or deliberately corrupt
IP frames

0x8038 dec-spanning DEC LANBridge Management

0x8039 dsm DEC DSM/DDP

0x8040 netbios DEC PATHWORKS DECnet NETBIOS Emulation

0x8041 msdos DEC Local Area System Transport

0x8042 etype-8042 DEC unassigned

0x809B appletalk Kinetics EtherTalk (AppleTalk over
Ethernet)

```

        0x80F3 aarp Kinetics AppleTalk Address Resolution
              Protocol (AARP)

        bpdusap      BPDUs SAP encapsulated packets
        bpdusnap     BPDUs SNAP encapsulated packets
        ipx-arp      IPX Advanced Research Projects Agency
                    (ARPA)
        ipx-non-arp  IPX non arp
        lacp         Link Aggregation Control Protocol(LACP)
                    encapsulated packets
        pagp         Port Aggregation Protocol(PAGP)
                    encapsulated packets
        vtp          VTP packets
        ";
    }

    typedef ip-protocol {
        type uint8{
            range "0..255";
        }
        description
            "The Internet Protocol (IP) is the principal communications
            protocol used for relaying datagrams (also known as network
            packets) across an internetwork using the Internet Protocol
            Suite.

            IP protocol number value is 0 to 255. It is an 8 bit field
            in the packet header";
        reference
            "IANA Protocol Numbers
            RFC5237 IANA Allocation Guidelines for the Protocol Field";
    }

    typedef igmp-code {
        //TODO: need more work. In NxOs, range is 0..15.
        // Could not match the IGMP with 0..15
        type uint8 /* {
            range "0..15";
        } */
        //IGMP v1 4 bits 0-15
        //IGMP v2 8bits. 0-
        //NXOS only support v1, but XR support v2.
        //

        description
            "Many of these IGMP types have a 'code' field. Here is
            the list of the types again with their assigned
            code fields."
    }

```

Type	Name	Reference
0x11	IGMP Membership Query	[RFC1112]
0x12	IGMPv1 Membership Report	[RFC1112]
0x13	DVMRP	[RFCDVMRP]
0x14	PIM version 1	[PIMv1]
0x15	Cisco Trace Messages	
0x16	IGMPv2 Membership Report	[RFC2236]
0x17	IGMPv2 Leave Group	[RFC2236]
0x1e	Multicast Traceroute Response	[Fenner]
0x1f	Multicast Traceroute	[Fenner]
0x22	IGMPv3 Membership Report	[RFC3376]

";

reference

"IANA Internet Group Management Protocol (IGMP) Type Numbers";

}

```
typedef icmp-type {
  type uint32 {
    range "0..255";
  }
  description
    "icmp-type is the Internet Control Message Protocol (ICMP)
    'type' field.
    The ICMP header starts after the IPv4 header. All ICMP
    packets will have an 8-byte header and variable-sized
    data section.
    The first 4 bytes of the header will be consistent.
    The first byte is for the ICMP type. The second byte is
    for the ICMP code.
    ICMP type is specified below
```

Type	Name	Reference
0	Echo Reply	[RFC792]
1	Unassigned	[JBP]
2	Unassigned	[JBP]
3	Destination Unreachable	[RFC792]
4	Source Quench	[RFC792]
5	Redirect	[RFC792]
6	Alternate Host Address	[JBP]
7	Unassigned	[JBP]
8	Echo	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Selection	[RFC1256]
11	Time Exceeded	[RFC792]
12	Parameter Problem	[RFC792]

```

13      Timestamp [RFC792]
14      Timestamp Reply [RFC792]
15      Information Request [RFC792]
16      Information Reply [RFC792]
17      Address Mask Request [RFC950]
18      Address Mask Reply [RFC950]
19      Reserved (for Security) [Solo]
20-29   Reserved (for Robustness Experiment) [ZSu]
30      Traceroute [RFC1393]
31      Datagram Conversion Error [RFC1475]
32      Mobile Host Redirect [David Johnson]
33      IPv6 Where-Are-You [Bill Simpson]
34      IPv6 I-Am-Here [Bill Simpson]
35      Mobile Registration Request [Bill Simpson]
36      Mobile Registration Reply [Bill Simpson]
37-255 Reserved [JBP]";
reference
  "RFC1700 ASSIGNED NUMBERS
  RFC792 Internet Control Message Protocol
  RFC4443 Internet Control Message Protocol (ICMPv6)
    for the Internet Protocol Version 6 (IPv6)
    Specification
  RFC2780 IANA Allocation Guidelines For Values In
    the Internet Protocol and Related Headers";
}

typedef icmp-code {
  type uint32 {
    range "0..255";
  }
  description
    "ICMP subtype to the given type.
    The ICMP header starts after the IPv4 header. All ICMP
    packets will have an 8-byte header and variable-sized
    data section.
    The first 4 bytes of the header will be consistent.
    The first byte is for the ICMP type. The second byte
    is for the ICMP code. ";
  reference "RFC2 INTERNET CONTROL MESSAGE PROTOCOL";
}

typedef vlan-identifier {
  type uint16 {
    range "1 .. 4095";
  }
  description
    "This type denotes a VLAN tag. ";
  reference

```

```
        "RFC3069 VLAN Aggregation for Efficient IP Address
          Allocation
          IEEE 802.1Q";
    }

    typedef time-to-live {
        type uint8 {
            range "0..255";
        }
        description "The TTL is an 8-bit field in IP header.
                    The maximum TTL value is 255.";
    }
}

<CODE ENDS>
```

15. Security Considerations

.

16. Open items from the previous revision

1. Are there any compatibility issues related to PFE ordering because a YANG user-order list is used instead of sequence IDs? This item is closely related to bullet item 3, see below.
2. Is an administrative function to test a packet against a specified SPF needed? The server would return an indication of permit or deny, and a leaf-list of the PFE entries that were evaluated. We believe that this addition would be valuable and have incorporated this suggestion into the "Additional Considerations" section. We expect to move it into the data model in the next revision.
3. Is the model applicable to multiple implementations - can other SPF models be accommodated? We have followed up with Juniper Yang experts, Kent Watsen and Phil Shafer, to review and check for applicability to Junos implementation. The initial feedback from Phil indicates that there do not seem to be any showstoppers and that the model does seem to be applicable. However, he suggested further scrutiny should occur. Kent identified additional Juniper experts to scrutinize the model more closely; so far no further comments have been received. We also followed up regarding whether there are other standardized models of SPFs, for example in conjunction with the Desktop Management Task Force's (DMTF) CIM (Common Information Model). SPF is not covered by the standardized portion of CIM, but there are vendor-specific

extensions by vendors. We inspected one such vendor specific model and found that in essence the same design patterns were used as in the model specified in this Internet Draft, with an SPF corresponding to an ordered list of rules with filters or matching criteria, and actions to be taken in response. It appears that mappings between the models can be accommodated in a straightforward manner.

17. Acknowledgements

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Louis Fourie, Dana Blair, Tula Kraiser, Patrick Gili, George Serpa, Martin Bjorklund, Kent Watsen, and Phil Shafer.

18. References

18.1. Normative References

- [RFC6020] Bjorklund, M., "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.

18.2. Informative References

- [if-config] Bjorklund, M., "A YANG Data Model for Interface Management", I-D draft-ietf-netmod-interfaces-cfg-12, July 2013.

Authors' Addresses

Lisa Huang
Cisco Systems
EMail: yihuan@cisco.com

Alexander Clemm
Cisco Systems
EMail: alex@cisco.com

Andy Bierman
YumaWorks

EMail: andy@yumaworks.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2014

M. Bjorklund
Tail-f Systems
July 4, 2013

IANA Address Family Numbers and Subsequent Address Family Identifiers
YANG Module
draft-ietf-netmod-iana-afn-safi-00

Abstract

This document defines the initial version of the iana-afn-safi 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 January 5, 2014.

Copyright Notice

Copyright (c) 2013 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 AFN and SAFI YANG Module	4
3. IANA Considerations	16
3.1. URI Registrations	17
3.2. YANG Module Registrations	17
4. Security Considerations	18
5. Normative References	19
Author's Address	20

1. Introduction

This document defines the initial version of the iana-afn-safi YANG module, for Address Family Numbers (AFN) and Subsequent Address Family Identifiers (SAFI).

The iana-afn-safi module reflects IANA's existing "Address Family Numbers" and "SAFI Values" registries.

Whenever a new address family number is added to the "Address Family Numbers" registry, the IANA-ADDRESS-FAMILY-NUMBERS-MIB and the iana-afn-safi YANG module are updated by IANA.

Whenever a new subsequent address family identifier is added to the "SAFI Values" registry, the iana-afn-safi YANG module is updated by IANA.

2. IANA Maintained AFN and SAFI YANG Module

<CODE BEGINS> file "iana-afn-safi.yang"

```
module iana-afn-safi {
  namespace "urn:ietf:params:xml:ns:yang:iana-afn-safi";
  prefix "ianaaf";

  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 provides two typedefs containing YANG
    definitions for the following IANA-registered enumerations:
```

- Address Family Numbers (AFN)
- Subsequent Address Family Identifiers (SAFI)

The latest revision of this YANG module can be obtained from the IANA web site.

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

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

This version of this YANG module is part of RFC 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 2013-07-04 {
```

```
description
  "Initial revision.";
reference
  "RFC XXXX: IANA Address Family Numbers and
    Subsequent Address Family Identifiers YANG Module";
}

typedef address-family {
  type enumeration {
    // value 0 is reserved by IANA
    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
        "HDLC (8-bit multidrop)";
    }
    enum bbn1822 {
      value "5";
      description
        "BBN 1822";
    }
    enum all802 {
      value "6";
      description
        "802 (includes all 802 media plus Ethernet 'canonical
        format')";
    }
    enum e163 {
      value "7";
      description
        "E.163";
    }
    enum e164 {
      value "8";
```

```
        description
        "E.164 (SMDS, FrameRelay, ATM)";
    }
    enum f69 {
        value "9";
        description
        "F.69 (Telex)";
    }
    enum x121 {
        value "10";
        description
        "X.121 (X.25, Frame Relay)";
    }
    enum ipx {
        value "11";
        description
        "IPX (Internetwork Packet Exchange)";
    }
    enum appletalk {
        value "12";
        description
        "Appletalk";
    }
    enum decnetIV {
        value "13";
        description
        "DECnet IV";
    }
    enum banyanVines {
        value "14";
        description
        "Banyan Vines";
    }
    enum e164withNsap {
        value "15";
        description
        "E.164 with NSAP format subaddress";
        reference
        "ATM Forum UNI 3.1";
    }
    enum dns {
        value "16";
        description
        "DNS (Domain Name System)";
    }
    enum distinguishedName {
        value "17";
        description
```

```
        "Distinguished Name (per X.500)";
    }
    enum asNumber {
        value "18";
        description
            "Autonomous System Number";
    }
    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";
    }
    // FIXME: This one is actually called "afi" in the MIB, but
    // that must be a mistake.
    enum l2vpn {
        value "25";
        description
            "AFI for L2VPN information";
        reference
            "RFC 4761: Virtual Private LAN Service (VPLS): Using BGP
            for Auto-Discovery and Signaling

            RFC 6074: Provisioning, Auto-Discovery, and Signaling in
```

```
        Layer 2 Virtual Private Networks (L2VPNs)";
    }
    enum mplsTpSectionEndpointIdentifier {
        value "26";
        description
            "MPLS-TP Section Endpoint Identifier";
        reference
            "draft-ietf-mpls-gach-adv";
    }
    enum mplsTpLspEndpointIdentifier {
        value "27";
        description
            "MPLS-TP LSP Endpoint Identifier";
        reference
            "draft-ietf-mpls-gach-adv";
    }
    enum mplsTpPseudowireEndpointIdentifier {
        value "28";
        description
            "MPLS-TP Pseudowire Endpoint Identifier";
        reference
            "draft-ietf-mpls-gach-adv";
    }
    enum eigrpCommonServiceFamily {
        value "16384";
        description
            "EIGRP Common Service Family";
    }
    enum eigrpIpv4ServiceFamily {
        value "16385";
        description
            "EIGRP IPv4 Service Family";
    }
    enum eigrpIpv6ServiceFamily {
        value "16386";
        description
            "EIGRP IPv6 Service Family";
    }
    enum lispCanonicalAddressFormat {
        value "16387";
        description
            "LISP Canonical Address Format (LCAF)";
    }
    enum bgpLs {
        value "16388";
        description
            "BGP-LS";
        reference
```



```
        "draft-ietf-idr-ls-distribution";
    }
    enum 48BitMac {
        value "16389";
        description
            "48-bit MAC";
        reference
            "draft-eastlake-rfc5342bis";
    }
    enum 64BitMac {
        value "16390";
        description
            "64-bit MAC";
        reference
            "draft-eastlake-rfc5342bis";
    }
    // value 65535 is reserved by IANA
}
description
    "This typedef is a YANG enumeration of IANA-registered address
    family numbers (AFN).";
reference
    "IANA Address Family Numbers registry.
    <http://www.iana.org/assignments/address-family-numbers>";
}

typedef subsequent-address-family {
    type enumeration {
        // value 0 is reserved by IANA
        enum nlriUnicast {
            value "1";
            description
                "Network Layer Reachability Information used for unicast
                forwarding";
            reference
                "RFC 4760: Multiprotocol Extensions for BGP-4";
        }
        enum nlriMulticast {
            value "2";
            description
                "Network Layer Reachability Information used for multicast
                forwarding";
            reference
                "RFC 4760: Multiprotocol Extensions for BGP-4";
        }
    }
    // value 3 is reserved by IANA
    enum nlriMpls {
        value "4";
```

```
    description
      "Network Layer Reachability Information (NLRI) with MPLS
      Labels";
    reference
      "RFC 3107: Carrying Label Information in BGP-4";
  }
  enum mcastVpn {
    value "5";
    description
      "MCAST-VPN";
    reference
      "RFC 6514: BGP Encodings and Procedures for Multicast in
      MPLS/BGP IP VPNs";
  }
  enum nlriDynamicMsPw {
    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: Dynamic Placement of Multi
      Segment Pseudowires";
  }
  enum encapsulation {
    value "7";
    description
      "Encapsulation SAFI";
    reference
      "RFC 5512: The BGP Encapsulation Subsequent Address Family
      Identifier (SAFI) and the BGP Tunnel Encapsulation
      Attribute";
  }
  enum tunnel {
    value "64";
    status "obsolete";
    description
      "Tunnel SAFI";
    reference
      "draft-nalawade-kapoor-tunnel-safi: BGP Tunnel SAFI";
  }
  enum vpls {
    value "65";
    description
      "Virtual Private LAN Service (VPLS)";
    reference
      "RFC 4761: Virtual Private LAN Service (VPLS): Using BGP
```

```
    for Auto-Discovery and Signaling

    RFC 6074: Provisioning, Auto-Discovery, and Signaling in
    Layer 2 Virtual Private Networks (L2VPNs)";
}
enum bgpMdt {
    value "66";
    description
        "BGP MDT SAFI";
    reference
        "RFC 6037: Cisco Systems' Solution for Multicast in
        BGP/MPLS IP VPNs";
}
enum bgp4over6 {
    value "67";
    description
        "BGP 4over6 SAFI";
    reference
        "RFC 5747: 4over6 Transit Solution Using IP Encapsulation
        and MP-BGP Extensions";
}
enum bgp6over4 {
    value "68";
    description
        "BGP 6over4 SAFI";
}
enum llVpnAutoDiscovery {
    value "69";
    description
        "Layer-1 VPN auto-discovery information";
    reference
        "RFC 5195: BGP-Based Auto-Discovery for Layer-1 VPNs";
}
enum mplsVpn {
    value "128";
    description
        "MPLS-labeled VPN address";
    reference
        "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs)";
}
enum multicastBgpMplsVpn {
    value "129";
    description
        "Multicast for BGP/MPLS IP Virtual Private Networks
        (VPNs)";
    reference
        "RFC 6513: Multicast in MPLS/BGP IP VPNs"
```

```
        RFC 6514: BGP Encodings and Procedures for Multicast in
        MPLS/BGP IP VPNs";
    }
    // values 130-131 are reserved by IANA
    enum routeTargetConstraints {
        value "132";
        description
            "Route Target constraints";
        reference
            "RFC 4684: Constrained Route Distribution for Border
            Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS)
            Internet Protocol (IP) Virtual Private Networks (VPNs)";
    }
    enum ipv4DissFlow {
        value "133";
        description
            "IPv4 dissemination of flow specification rules";
        reference
            "RFC 5575: Dissemination of Flow Specification Rules";
    }
    enum vpnv4DissFlow {
        value "134";
        description
            "VPNv4 dissemination of flow specification rules";
        reference
            "RFC 5575: Dissemination of Flow Specification Rules";
    }
    // values 135-139 are reserved by IANA
    enum vpnAutoDiscovery {
        value "140";
        status "obsolete";
        description
            "VPN auto-discovery";
        reference
            "draft-ietf-l3vpn-bgpvpn-auto: Using BGP as an
            Auto-Discovery Mechanism for VR-based Layer-3 VPNs";
    }
    // values 141-240 are reserved by IANA
    enum private241 {
        value "241";
        description
            "Reserved for Private Use";
        reference
            "RFC 4760: Multiprotocol Extensions for BGP-4";
    }
    enum private242 {
        value "242";
        description
```

```
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private243 {
    value "243";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private244 {
    value "244";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private245 {
    value "245";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private246 {
    value "246";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private247 {
    value "247";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private248 {
    value "248";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private249 {
    value "249";
```

```
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private250 {
    value "250";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private251 {
    value "251";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private252 {
    value "252";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private253 {
    value "253";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
enum private254 {
    value "254";
    description
        "Reserved for Private Use";
    reference
        "RFC 4760: Multiprotocol Extensions for BGP-4";
}
// value 255 is reserved by IANA
}
description
    "This typedef is a YANG enumeration of IANA-registered
    subsequent address family identifiers (SAFI).";
reference
    "IANA SAFI Values registry.
    <http://www.iana.org/assignments/safi-namespace>";
```

```
}  
}
```

```
<CODE ENDS>
```

3. IANA Considerations

This document defines the initial version of the IANA-maintained iana-afn-safi YANG module.

IANA is requested to extend the registries "Address Family Numbers" and "SAFI Values" with a "Name" column. IANA is also requested to add this new Note to these registries:

The name of an entry in this registry must be a legal SMIV2 enumeration label.

The existing entries in the "Address Family Numbers" registry should get their names from the corresponding "enum" statement in the "address-family" typedef.

The existing entries in the "SAFI Values" registry should get their names from the corresponding "enum" statement in the "subsequent-address-family" typedef.

The iana-afn-safi module is intended to reflect the "Address Family Numbers" and "SAFI Values" registries. When an AFN or SAFI is added to these registries, a new "enum" statement must be added to the "address-family" or "subsequent-address-family" typedefs. The name of the "enum" is the value of the "Name" column in the registry.

The following substatements to the "enum" statement should be defined:

"value": Replicate the value from the registry.

"status": Include only if a registration has been deprecated (use the value "deprecated") or obsoleted (use the value "obsolete").

"description": Replicate the description from the registry, if any.

"reference": Replicate the reference from the registry, if any, and add the title of the document.

If a parameter is marked as "reserved" in these registries, no "enum" statement is added to the corresponding typedef. Instead a comment is added, on the form:

```
// value NN is reserved by XX
```

Unassigned values are not present in the module.

When the iana-afn-safi YANG module is updated, a new "revision" statement must be added.

IANA is requested to add this new Note to the "Address Family Numbers" and "SAFI Values" registries:

When this registry is modified, the YANG module iana-afn-safi must be updated as defined in RFC XXXX.

The Reference text in the "Address Family Numbers" registry needs to be updated as:

OLD:

[RFC2453][RFC2858]

NEW:

[RFC2453][RFC2858][RFCXXXX]

The Reference text in the "SAFI Values" registry needs to be updated as:

OLD:

[RFC4760]

NEW:

[RFC4760][RFCXXXX]

3.1. URI Registrations

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-afn-safi

Registrant Contact: IANA.

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

3.2. YANG Module Registrations

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

name:	iana-afn-safi
namespace:	urn:ietf:params:xml:ns:yang:iana-afn-safi
prefix:	ianaaf
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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2014

M. Bjorklund
Tail-f Systems
July 4, 2013

IANA Interface Type YANG Module
draft-ietf-netmod-iana-if-type-07

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 January 5, 2014.

Copyright Notice

Copyright (c) 2013 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	36
3.1. URI Registrations	37
3.2. YANG Module Registrations	37
4. Security Considerations	38
5. Normative References	39
Author's Address	40

1. Introduction

This document defines the initial version of the iana-if-type YANG module for interface type definitions.

The iana-if-type module reflects IANA's existing "ifType definitions" registry. The latest revision of the module can be obtained from the IANA web site.

Whenever a new interface type is added to the "ifType definitions" registry, the IANAifType-MIB and the iana-if-type YANG module are updated by IANA.

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 ianaift;

  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.

    This YANG module is maintained by IANA, and reflects the
    'ifType definitions' registry.

    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 2013-07-04 {
    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 "ds1" {
        value 18;
        description
            "DS1-MIB";
        reference
            "RFC 4805 - Definitions of Managed Objects for the
             DS1, J1, E1, DS2, and E2 Interface Types";
    }
    enum "e1" {
        value 19;
        status obsolete;
    }
```

```
    description
      "Obsolete see DS1-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 "iso8802211c" {
  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 "dls" {
        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 SMIPv2 (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 "al2MppSwitch" {
  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 "frf16MfrBundle" {
        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 Communications";
    }
```

```
}
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 "mvl" {
    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/OBSOLETED - 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";
    }
    enum "vmwareVirtualNic" {
        value 258;
        description
            "VMware Virtual Network Interface";
    }
    enum "ieee802154" {
        value 259;
        description
            "IEEE 802.15.4 WPAN interface";
        reference
            "IEEE 802.15.4-2006";
    }
    enum "otnOdu" {
        value 260;
        description
            "OTN Optical Data Unit";
    }
    enum "otnOtu" {
        value 261;
        description
            "OTN Optical channel Transport Unit";
    }
    enum "ifVfiType" {
```

```
        value 262;
        description
            "VPLS Forwarding Instance Interface Type";
    }
    enum "g9981" {
        value 263;
        description
            "G.998.1 bonded interface";
    }
    enum "g9982" {
        value 264;
        description
            "G.998.2 bonded interface";
    }
    enum "g9983" {
        value 265;
        description
            "G.998.3 bonded interface";
    }
    enum "aluEpon" {
        value 266;
        description
            "Ethernet Passive Optical Networks (E-PON)";
    }
    enum "aluEponOnu" {
        value 267;
        description
            "EPON Optical Network Unit";
    }
    enum "aluEponPhysicalUni" {
        value 268;
        description
            "EPON physical User to Network interface";
    }
    enum "aluEponLogicalLink" {
        value 269;
        description
            "The emulation of a point-to-point link over the EPON
            layer";
    }
    enum "aluGponOnu" {
        value 270;
        description
            "GPON Optical Network Unit";
        reference
            "ITU-T G.984.2";
    }
    enum "aluGponPhysicalUni" {
```

```
        value 271;
        description
            "GPON physical User to Network interface";
        reference
            "ITU-T G.984.2";
    }
    enum "vmwareNicTeam" {
        value 272;
        description
            "VMware NIC Team";
    }
    // value 273 reserved by IANA
}
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).";
reference
    "IANA ifType definitions registry.
    <http://www.iana.org/assignments/smi-numbers>";
}
}
```

<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. The name of the "enum" is the same as the corresponding enumeration in the IANAifType-MIB. The following substatements to the "enum" statement should be defined:

"value": Replicate the value from the registry.

"status": Include only if a registration has been deprecated (use the value "deprecated") or obsoleted (use the value "obsolete").

"description": Replicate the description from the registry, if any.

"reference": Replicate the reference from the registry, if any, and add the title of the document.

If an interface type is marked as "reserved" in the "ifType definitions" registry, no "enum" statement is added to the "iana-if-type" typedef. Instead a comment is added, on the form:

```
// value NN is reserved by XX
```

Unassigned values are not present in the module.

When the iana-if-type YANG module is updated, a new "revision" statement must be added.

IANA is requested to add this new Note to the "ifType definitions" registry:

When this registry is modified, the YANG module iana-if-type must be updated as defined in RFC XXXX.

The Reference text in the "ifType definitions" registry needs to be updated as:

OLD:

[RFC1213][RFC2863]

NEW:

[RFC1213][RFC2863][RFCXXXX]

3.1. URI Registrations

This document registers a URIs 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: IANA.

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

3.2. YANG Module Registrations

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

name:	iana-if-type
namespace:	urn:ietf:params:xml:ns:yang:iana-if-type
prefix:	ianaift
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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2013

J. Lange
GE Digital Energy
July 09, 2012

IANA Timezone Database YANG Module
draft-ietf-netmod-iana-timezones-00

Abstract

This document defines the initial version of the iana-timezones 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 January 10, 2013.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. IANA Maintained Timezones YANG Module	3
3. IANA Considerations	38
4. Security Considerations	39
5. Normative References	39
Author's Address	40

1. Introduction

This document defines the initial version of the iana-timezones YANG module for timezone configuration.

The iana-timezones module reflects IANA's existing "timezone database". The latest revision of the module can be obtained from the IANA web site.

Whenever a new timezone name is added to the IANA "timezone database", the iana-timezones module is updated by IANA.

2. IANA Maintained Timezones YANG Module

```
<CODE BEGINS> file "iana-timezones@2012-07-09.yang"
module iana-timezones {
  namespace "urn:ietf:params:xml:ns:yang:iana-timezones";
  prefix ianatz;

  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-timezone typedef, which
    contains YANG definitions for IANA-registered timezones.

    This YANG module is maintained by IANA, and reflects the
    IANA Time Zone Database.
    (http://www.iana.org/time-zones)

    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.";

```
revision 2012-07-09 {
  description
    "Initial revision. Using IANA Time Zone Data v. 2012c
    (Released 2012-03-27)";
  reference "RFC XXXX: TITLE";
}
typedef iana-timezone {
  description
    "A timezone location as defined by the IANA timezone
    database (http://www.iana.org/time-zones)";
  type enumeration {
    enum "Europe/Andorra" {
      value 0;
    }
    enum "Asia/Dubai" {
      value 1;
    }
    enum "Asia/Kabul" {
      value 2;
    }
    enum "America/Antigua" {
      value 3;
    }
    enum "America/Anguilla" {
      value 4;
    }
    enum "Europe/Tirane" {
      value 5;
    }
    enum "Asia/Yerevan" {
      value 6;
    }
    enum "Africa/Luanda" {
      value 7;
    }
    enum "Antarctica/McMurdo" {
      value 8;
      description
        "McMurdo Station, Ross Island";
    }
    enum "Antarctica/South_Pole" {
      value 9;
    }
  }
}
```

```
        description
            "Amundsen-Scott Station, South Pole";
    }
    enum "Antarctica/Rothera" {
        value 10;
        description
            "Rothera Station, Adelaide Island";
    }
    enum "Antarctica/Palmer" {
        value 11;
        description
            "Palmer Station, Anvers Island";
    }
    enum "Antarctica/Mawson" {
        value 12;
        description
            "Mawson Station, Holme Bay";
    }
    enum "Antarctica/Davis" {
        value 13;
        description
            "Davis Station, Vestfold Hills";
    }
    enum "Antarctica/Casey" {
        value 14;
        description
            "Casey Station, Bailey Peninsula";
    }
    enum "Antarctica/Vostok" {
        value 15;
        description
            "Vostok Station, Lake Vostok";
    }
    enum "Antarctica/DumontDUrville" {
        value 16;
        description
            "Dumont-d'Urville Station, Terre Adelie";
    }
    enum "Antarctica/Syowa" {
        value 17;
        description
            "Syowa Station, E Ongul I";
    }
    enum "Antarctica/Macquarie" {
        value 18;
        description
            "Macquarie Island Station, Macquarie Island";
    }
}
```

```
enum "America/Argentina/Buenos_Aires" {
    value 19;
    description
        "Buenos Aires (BA, CF)";
}
enum "America/Argentina/Cordoba" {
    value 20;
    description
        "most locations (CB, CC, CN, ER, FM, MN, SE, SF)";
}
enum "America/Argentina/Salta" {
    value 21;
    description
        "(SA, LP, NQ, RN)";
}
enum "America/Argentina/Jujuy" {
    value 22;
    description
        "Jujuy (JY)";
}
enum "America/Argentina/Tucuman" {
    value 23;
    description
        "Tucuman (TM)";
}
enum "America/Argentina/Catamarca" {
    value 24;
    description
        "Catamarca (CT), Chubut (CH)";
}
enum "America/Argentina/La_Rioja" {
    value 25;
    description
        "La Rioja (LR)";
}
enum "America/Argentina/San_Juan" {
    value 26;
    description
        "San Juan (SJ)";
}
enum "America/Argentina/Mendoza" {
    value 27;
    description
        "Mendoza (MZ)";
}
enum "America/Argentina/San_Luis" {
    value 28;
    description
```

```
        "San Luis (SL)";
    }
    enum "America/Argentina/Rio_Gallegos" {
        value 29;
        description
            "Santa Cruz (SC)";
    }
    enum "America/Argentina/Ushuaia" {
        value 30;
        description
            "Tierra del Fuego (TF)";
    }
    enum "Pacific/Pago_Pago" {
        value 31;
    }
    enum "Europe/Vienna" {
        value 32;
    }
    enum "Australia/Lord_Howe" {
        value 33;
        description
            "Lord Howe Island";
    }
    enum "Australia/Hobart" {
        value 34;
        description
            "Tasmania - most locations";
    }
    enum "Australia/Currie" {
        value 35;
        description
            "Tasmania - King Island";
    }
    enum "Australia/Melbourne" {
        value 36;
        description
            "Victoria";
    }
    enum "Australia/Sydney" {
        value 37;
        description
            "New South Wales - most locations";
    }
    enum "Australia/Broken_Hill" {
        value 38;
        description
            "New South Wales - Yancowinna";
    }
}
```

```
enum "Australia/Brisbane" {
  value 39;
  description
    "Queensland - most locations";
}
enum "Australia/Lindeman" {
  value 40;
  description
    "Queensland - Holiday Islands";
}
enum "Australia/Adelaide" {
  value 41;
  description
    "South Australia";
}
enum "Australia/Darwin" {
  value 42;
  description
    "Northern Territory";
}
enum "Australia/Perth" {
  value 43;
  description
    "Western Australia - most locations";
}
enum "Australia/Eucla" {
  value 44;
  description
    "Western Australia - Eucla area";
}
enum "America/Aruba" {
  value 45;
}
enum "Europe/Mariehamn" {
  value 46;
}
enum "Asia/Baku" {
  value 47;
}
enum "Europe/Sarajevo" {
  value 48;
}
enum "America/Barbados" {
  value 49;
}
enum "Asia/Dhaka" {
  value 50;
}
```

```
enum "Europe/Brussels" {
    value 51;
}
enum "Africa/Ouagadougou" {
    value 52;
}
enum "Europe/Sofia" {
    value 53;
}
enum "Asia/Bahrain" {
    value 54;
}
enum "Africa/Bujumbura" {
    value 55;
}
enum "Africa/Porto-Novo" {
    value 56;
}
enum "America/St_Barthelemy" {
    value 57;
}
enum "Atlantic/Bermuda" {
    value 58;
}
enum "Asia/Brunei" {
    value 59;
}
enum "America/La_Paz" {
    value 60;
}
enum "America/Kralendijk" {
    value 61;
}
enum "America/Noronha" {
    value 62;
    description
        "Atlantic islands";
}
enum "America/Belem" {
    value 63;
    description
        "Amapa, E Para";
}
enum "America/Fortaleza" {
    value 64;
    description
        "NE Brazil (MA, PI, CE, RN, PB)";
}
```

```
enum "America/Recife" {
  value 65;
  description
    "Pernambuco";
}
enum "America/Araguaina" {
  value 66;
  description
    "Tocantins";
}
enum "America/Maceio" {
  value 67;
  description
    "Alagoas, Sergipe";
}
enum "America/Bahia" {
  value 68;
  description
    "Bahia";
}
enum "America/Sao_Paulo" {
  value 69;
  description
    "S & SE Brazil (GO, DF, MG, ES, RJ, SP, PR, SC, RS)";
}
enum "America/Campo_Grande" {
  value 70;
  description
    "Mato Grosso do Sul";
}
enum "America/Cuiaba" {
  value 71;
  description
    "Mato Grosso";
}
enum "America/Santarem" {
  value 72;
  description
    "W Para";
}
enum "America/Porto_Velho" {
  value 73;
  description
    "Rondonia";
}
enum "America/Boa_Vista" {
  value 74;
  description
```



```
        "Roraima";
    }
    enum "America/Manaus" {
        value 75;
        description
            "E Amazonas";
    }
    enum "America/Eirunepe" {
        value 76;
        description
            "W Amazonas";
    }
    enum "America/Rio_Branco" {
        value 77;
        description
            "Acre";
    }
    enum "America/Nassau" {
        value 78;
    }
    enum "Asia/Thimphu" {
        value 79;
    }
    enum "Africa/Gaborone" {
        value 80;
    }
    enum "Europe/Minsk" {
        value 81;
    }
    enum "America/Belize" {
        value 82;
    }
    enum "America/St_Johns" {
        value 83;
        description
            "Newfoundland Time, including SE Labrador";
    }
    enum "America/Halifax" {
        value 84;
        description
            "Atlantic Time - Nova Scotia (most places), PEI";
    }
    enum "America/Glace_Bay" {
        value 85;
        description
            "Atlantic Time - Nova Scotia - places that did not observe
            DST 1966-1971";
    }
}
```

```
enum "America/Moncton" {
  value 86;
  description
    "Atlantic Time - New Brunswick";
}
enum "America/Goose_Bay" {
  value 87;
  description
    "Atlantic Time - Labrador - most locations";
}
enum "America/Blanc-Sablon" {
  value 88;
  description
    "Atlantic Standard Time - Quebec - Lower North Shore";
}
enum "America/Montreal" {
  value 89;
  description
    "Eastern Time - Quebec - most locations";
}
enum "America/Toronto" {
  value 90;
  description
    "Eastern Time - Ontario - most locations";
}
enum "America/Nipigon" {
  value 91;
  description
    "Eastern Time - Ontario & Quebec - places that did not
    observe DST 1967-1973";
}
enum "America/Thunder_Bay" {
  value 92;
  description
    "Eastern Time - Thunder Bay, Ontario";
}
enum "America/Iqaluit" {
  value 93;
  description
    "Eastern Time - east Nunavut - most locations";
}
enum "America/Pangnirtung" {
  value 94;
  description
    "Eastern Time - Pangnirtung, Nunavut";
}
enum "America/Resolute" {
  value 95;
```

```
    description
      "Central Standard Time - Resolute, Nunavut";
  }
  enum "America/Atikokan" {
    value 96;
    description
      "Eastern Standard Time - Atikokan, Ontario and Southampton I,
      Nunavut";
  }
  enum "America/Rankin_Inlet" {
    value 97;
    description
      "Central Time - central Nunavut";
  }
  enum "America/Winnipeg" {
    value 98;
    description
      "Central Time - Manitoba & west Ontario";
  }
  enum "America/Rainy_River" {
    value 99;
    description
      "Central Time - Rainy River & Fort Frances, Ontario";
  }
  enum "America/Regina" {
    value 100;
    description
      "Central Standard Time - Saskatchewan - most locations";
  }
  enum "America/Swift_Current" {
    value 101;
    description
      "Central Standard Time - Saskatchewan - midwest";
  }
  enum "America/Edmonton" {
    value 102;
    description
      "Mountain Time - Alberta, east British Columbia & west
      Saskatchewan";
  }
  enum "America/Cambridge_Bay" {
    value 103;
    description
      "Mountain Time - west Nunavut";
  }
  enum "America/Yellowknife" {
    value 104;
    description
```

```
        "Mountain Time - central Northwest Territories";
    }
    enum "America/Inuvik" {
        value 105;
        description
            "Mountain Time - west Northwest Territories";
    }
    enum "America/Creston" {
        value 106;
        description
            "Mountain Standard Time - Creston, British Columbia";
    }
    enum "America/Dawson_Creek" {
        value 107;
        description
            "Mountain Standard Time - Dawson Creek & Fort Saint John,
            British Columbia";
    }
    enum "America/Vancouver" {
        value 108;
        description
            "Pacific Time - west British Columbia";
    }
    enum "America/Whitehorse" {
        value 109;
        description
            "Pacific Time - south Yukon";
    }
    enum "America/Dawson" {
        value 110;
        description
            "Pacific Time - north Yukon";
    }
    enum "Indian/Cocos" {
        value 111;
    }
    enum "Africa/Kinshasa" {
        value 112;
        description
            "west Dem. Rep. of Congo";
    }
    enum "Africa/Lubumbashi" {
        value 113;
        description
            "east Dem. Rep. of Congo";
    }
    enum "Africa/Bangui" {
        value 114;
```

```
}
enum "Africa/Brazzaville" {
    value 115;
}
enum "Europe/Zurich" {
    value 116;
}
enum "Africa/Abidjan" {
    value 117;
}
enum "Pacific/Rarotonga" {
    value 118;
}
enum "America/Santiago" {
    value 119;
    description
        "most locations";
}
enum "Pacific/Easter" {
    value 120;
    description
        "Easter Island & Sala y Gomez";
}
enum "Africa/Douala" {
    value 121;
}
enum "Asia/Shanghai" {
    value 122;
    description
        "east China - Beijing, Guangdong, Shanghai, etc.";
}
enum "Asia/Harbin" {
    value 123;
    description
        "Heilongjiang (except Mohe), Jilin";
}
enum "Asia/Chongqing" {
    value 124;
    description
        "central China - Sichuan, Yunnan, Guangxi, Shaanxi, Guizhou,
        etc.";
}
enum "Asia/Urumqi" {
    value 125;
    description
        "most of Tibet & Xinjiang";
}
enum "Asia/Kashgar" {
```

```
        value 126;
        description
            "west Tibet & Xinjiang";
    }
    enum "America/Bogota" {
        value 127;
    }
    enum "America/Costa_Rica" {
        value 128;
    }
    enum "America/Havana" {
        value 129;
    }
    enum "Atlantic/Cape_Verde" {
        value 130;
    }
    enum "America/Curacao" {
        value 131;
    }
    enum "Indian/Christmas" {
        value 132;
    }
    enum "Asia/Nicosia" {
        value 133;
    }
    enum "Europe/Prague" {
        value 134;
    }
    enum "Europe/Berlin" {
        value 135;
    }
    enum "Africa/Djibouti" {
        value 136;
    }
    enum "Europe/Copenhagen" {
        value 137;
    }
    enum "America/Dominica" {
        value 138;
    }
    enum "America/Santo_Domingo" {
        value 139;
    }
    enum "Africa/Algiers" {
        value 140;
    }
    enum "America/Guayaquil" {
        value 141;
```

```
        description
            "mainland";
    }
    enum "Pacific/Galapagos" {
        value 142;
        description
            "Galapagos Islands";
    }
    enum "Europe/Tallinn" {
        value 143;
    }
    enum "Africa/Cairo" {
        value 144;
    }
    enum "Africa/El_Aaiun" {
        value 145;
    }
    enum "Africa/Asmara" {
        value 146;
    }
    enum "Europe/Madrid" {
        value 147;
        description
            "mainland";
    }
    enum "Africa/Ceuta" {
        value 148;
        description
            "Ceuta & Melilla";
    }
    enum "Atlantic/Canary" {
        value 149;
        description
            "Canary Islands";
    }
    enum "Africa/Addis_Ababa" {
        value 150;
    }
    enum "Europe/Helsinki" {
        value 151;
    }
    enum "Pacific/Fiji" {
        value 152;
    }
    enum "Atlantic/Stanley" {
        value 153;
    }
    enum "Pacific/Chuuk" {
```

```
        value 154;
        description
            "Chuuk (Truk) and Yap";
    }
    enum "Pacific/Pohnpei" {
        value 155;
        description
            "Pohnpei (Ponape)";
    }
    enum "Pacific/Kosrae" {
        value 156;
        description
            "Kosrae";
    }
    enum "Atlantic/Faroe" {
        value 157;
    }
    enum "Europe/Paris" {
        value 158;
    }
    enum "Africa/Libreville" {
        value 159;
    }
    enum "Europe/London" {
        value 160;
    }
    enum "America/Grenada" {
        value 161;
    }
    enum "Asia/Tbilisi" {
        value 162;
    }
    enum "America/Cayenne" {
        value 163;
    }
    enum "Europe/Guernsey" {
        value 164;
    }
    enum "Africa/Accra" {
        value 165;
    }
    enum "Europe/Gibraltar" {
        value 166;
    }
    enum "America/Godthab" {
        value 167;
        description
            "most locations";
    }
```



```
}
enum "America/Danmarkshavn" {
  value 168;
  description
    "east coast, north of Scoresbysund";
}
enum "America/Scoresbysund" {
  value 169;
  description
    "Scoresbysund / Ittoqqortoormiit";
}
enum "America/Thule" {
  value 170;
  description
    "Thule / Pituffik";
}
enum "Africa/Banjul" {
  value 171;
}
enum "Africa/Conakry" {
  value 172;
}
enum "America/Guadeloupe" {
  value 173;
}
enum "Africa/Malabo" {
  value 174;
}
enum "Europe/Athens" {
  value 175;
}
enum "Atlantic/South_Georgia" {
  value 176;
}
enum "America/Guatemala" {
  value 177;
}
enum "Pacific/Guam" {
  value 178;
}
enum "Africa/Bissau" {
  value 179;
}
enum "America/Guyana" {
  value 180;
}
enum "Asia/Hong_Kong" {
  value 181;
```

```
}
enum "America/Tegucigalpa" {
    value 182;
}
enum "Europe/Zagreb" {
    value 183;
}
enum "America/Port-au-Prince" {
    value 184;
}
enum "Europe/Budapest" {
    value 185;
}
enum "Asia/Jakarta" {
    value 186;
    description
        "Java & Sumatra";
}
enum "Asia/Pontianak" {
    value 187;
    description
        "west & central Borneo";
}
enum "Asia/Makassar" {
    value 188;
    description
        "east & south Borneo, Sulawesi (Celebes), Bali, Nusa
        Tenggara, west Timor";
}
enum "Asia/Jayapura" {
    value 189;
    description
        "west New Guinea (Irian Jaya) & Maluku (Moluccas)";
}
enum "Europe/Dublin" {
    value 190;
}
enum "Asia/Jerusalem" {
    value 191;
}
enum "Europe/Isle_of_Man" {
    value 192;
}
enum "Asia/Kolkata" {
    value 193;
}
enum "Indian/Chagos" {
    value 194;
```

```
}
enum "Asia/Baghdad" {
    value 195;
}
enum "Asia/Tehran" {
    value 196;
}
enum "Atlantic/Reykjavik" {
    value 197;
}
enum "Europe/Rome" {
    value 198;
}
enum "Europe/Jersey" {
    value 199;
}
enum "America/Jamaica" {
    value 200;
}
enum "Asia/Amman" {
    value 201;
}
enum "Asia/Tokyo" {
    value 202;
}
enum "Africa/Nairobi" {
    value 203;
}
enum "Asia/Bishkek" {
    value 204;
}
enum "Asia/Phnom_Penh" {
    value 205;
}
enum "Pacific/Tarawa" {
    value 206;
    description
        "Gilbert Islands";
}
enum "Pacific/Enderbury" {
    value 207;
    description
        "Phoenix Islands";
}
enum "Pacific/Kiritimati" {
    value 208;
    description
        "Line Islands";
}
```

```
}
enum "Indian/Comoro" {
  value 209;
}
enum "America/St_Kitts" {
  value 210;
}
enum "Asia/Pyongyang" {
  value 211;
}
enum "Asia/Seoul" {
  value 212;
}
enum "Asia/Kuwait" {
  value 213;
}
enum "America/Cayman" {
  value 214;
}
enum "Asia/Almaty" {
  value 215;
  description
    "most locations";
}
enum "Asia/Qyzylorda" {
  value 216;
  description
    "Qyzylorda (Kyzylorda, Kzyl-Orda)";
}
enum "Asia/Aqtobe" {
  value 217;
  description
    "Aqtobe (Aktobe)";
}
enum "Asia/Aqtau" {
  value 218;
  description
    "Atyrau (Atirau, Gur'yev), Mangghystau (Mankistau)";
}
enum "Asia/Oral" {
  value 219;
  description
    "West Kazakhstan";
}
enum "Asia/Vientiane" {
  value 220;
}
enum "Asia/Beirut" {
```

```
    value 221;
  }
  enum "America/St_Lucia" {
    value 222;
  }
  enum "Europe/Vaduz" {
    value 223;
  }
  enum "Asia/Colombo" {
    value 224;
  }
  enum "Africa/Monrovia" {
    value 225;
  }
  enum "Africa/Maseru" {
    value 226;
  }
  enum "Europe/Vilnius" {
    value 227;
  }
  enum "Europe/Luxembourg" {
    value 228;
  }
  enum "Europe/Riga" {
    value 229;
  }
  enum "Africa/Tripoli" {
    value 230;
  }
  enum "Africa/Casablanca" {
    value 231;
  }
  enum "Europe/Monaco" {
    value 232;
  }
  enum "Europe/Chisinau" {
    value 233;
  }
  enum "Europe/Podgorica" {
    value 234;
  }
  enum "America/Marigot" {
    value 235;
  }
  enum "Indian/Antananarivo" {
    value 236;
  }
  enum "Pacific/Majuro" {
```

```
        value 237;
        description
            "most locations";
    }
    enum "Pacific/Kwajalein" {
        value 238;
        description
            "Kwajalein";
    }
    enum "Europe/Skopje" {
        value 239;
    }
    enum "Africa/Bamako" {
        value 240;
    }
    enum "Asia/Rangoon" {
        value 241;
    }
    enum "Asia/Ulaanbaatar" {
        value 242;
        description
            "most locations";
    }
    enum "Asia/Hovd" {
        value 243;
        description
            "Bayan-Olgii, Govi-Altai, Hovd, Uvs, Zavkhan";
    }
    enum "Asia/Choibalsan" {
        value 244;
        description
            "Dornod, Sukhbaatar";
    }
    enum "Asia/Macau" {
        value 245;
    }
    enum "Pacific/Saipan" {
        value 246;
    }
    enum "America/Martinique" {
        value 247;
    }
    enum "Africa/Nouakchott" {
        value 248;
    }
    enum "America/Montserrat" {
        value 249;
    }
}
```

```
enum "Europe/Malta" {  
    value 250;  
}  
enum "Indian/Mauritius" {  
    value 251;  
}  
enum "Indian/Maldives" {  
    value 252;  
}  
enum "Africa/Blantyre" {  
    value 253;  
}  
enum "America/Mexico_City" {  
    value 254;  
    description  
        "Central Time - most locations";  
}  
enum "America/Cancun" {  
    value 255;  
    description  
        "Central Time - Quintana Roo";  
}  
enum "America/Merida" {  
    value 256;  
    description  
        "Central Time - Campeche, Yucatan";  
}  
enum "America/Monterrey" {  
    value 257;  
    description  
        "Mexican Central Time - Coahuila, Durango, Nuevo Leon,  
        Tamaulipas away from US border";  
}  
enum "America/Matamoros" {  
    value 258;  
    description  
        "US Central Time - Coahuila, Durango, Nuevo Leon, Tamaulipas  
        near US border";  
}  
enum "America/Mazatlan" {  
    value 259;  
    description  
        "Mountain Time - S Baja, Nayarit, Sinaloa";  
}  
enum "America/Chihuahua" {  
    value 260;  
    description  
        "Mexican Mountain Time - Chihuahua away from US border";
```

```
}
enum "America/Ojinaga" {
  value 261;
  description
    "US Mountain Time - Chihuahua near US border";
}
enum "America/Hermosillo" {
  value 262;
  description
    "Mountain Standard Time - Sonora";
}
enum "America/Tijuana" {
  value 263;
  description
    "US Pacific Time - Baja California near US border";
}
enum "America/Santa_Isabel" {
  value 264;
  description
    "Mexican Pacific Time - Baja California away from US border";
}
enum "America/Bahia_Banderas" {
  value 265;
  description
    "Mexican Central Time - Bahia de Banderas";
}
enum "Asia/Kuala_Lumpur" {
  value 266;
  description
    "peninsular Malaysia";
}
enum "Asia/Kuching" {
  value 267;
  description
    "Sabah & Sarawak";
}
enum "Africa/Maputo" {
  value 268;
}
enum "Africa/Windhoek" {
  value 269;
}
enum "Pacific/Noumea" {
  value 270;
}
enum "Africa/Niamey" {
  value 271;
}
```



```
enum "Pacific/Norfolk" {
    value 272;
}
enum "Africa/Lagos" {
    value 273;
}
enum "America/Managua" {
    value 274;
}
enum "Europe/Amsterdam" {
    value 275;
}
enum "Europe/Oslo" {
    value 276;
}
enum "Asia/Kathmandu" {
    value 277;
}
enum "Pacific/Nauru" {
    value 278;
}
enum "Pacific/Niue" {
    value 279;
}
enum "Pacific/Auckland" {
    value 280;
    description
        "most locations";
}
enum "Pacific/Chatham" {
    value 281;
    description
        "Chatham Islands";
}
enum "Asia/Muscat" {
    value 282;
}
enum "America/Panama" {
    value 283;
}
enum "America/Lima" {
    value 284;
}
enum "Pacific/Tahiti" {
    value 285;
    description
        "Society Islands";
}
```

```
enum "Pacific/Marquesas" {  
    value 286;  
    description  
        "Marquesas Islands";  
}  
enum "Pacific/Gambier" {  
    value 287;  
    description  
        "Gambier Islands";  
}  
enum "Pacific/Port_Moresby" {  
    value 288;  
}  
enum "Asia/Manila" {  
    value 289;  
}  
enum "Asia/Karachi" {  
    value 290;  
}  
enum "Europe/Warsaw" {  
    value 291;  
}  
enum "America/Miquelon" {  
    value 292;  
}  
enum "Pacific/Pitcairn" {  
    value 293;  
}  
enum "America/Puerto_Rico" {  
    value 294;  
}  
enum "Asia/Gaza" {  
    value 295;  
    description  
        "Gaza Strip";  
}  
enum "Asia/Hebron" {  
    value 296;  
    description  
        "West Bank";  
}  
enum "Europe/Lisbon" {  
    value 297;  
    description  
        "mainland";  
}  
enum "Atlantic/Madeira" {  
    value 298;
```

```
        description
            "Madeira Islands";
    }
    enum "Atlantic/Azores" {
        value 299;
        description
            "Azores";
    }
    enum "Pacific/Palau" {
        value 300;
    }
    enum "America/Asuncion" {
        value 301;
    }
    enum "Asia/Qatar" {
        value 302;
    }
    enum "Indian/Reunion" {
        value 303;
    }
    enum "Europe/Bucharest" {
        value 304;
    }
    enum "Europe/Belgrade" {
        value 305;
    }
    enum "Europe/Kaliningrad" {
        value 306;
        description
            "Moscow-01 - Kaliningrad";
    }
    enum "Europe/Moscow" {
        value 307;
        description
            "Moscow+00 - west Russia";
    }
    enum "Europe/Volgograd" {
        value 308;
        description
            "Moscow+00 - Caspian Sea";
    }
    enum "Europe/Samara" {
        value 309;
        description
            "Moscow+00 - Samara, Udmurtia";
    }
    enum "Asia/Yekaterinburg" {
        value 310;
```

```
        description
            "Moscow+02 - Urals";
    }
    enum "Asia/Omsk" {
        value 311;
        description
            "Moscow+03 - west Siberia";
    }
    enum "Asia/Novosibirsk" {
        value 312;
        description
            "Moscow+03 - Novosibirsk";
    }
    enum "Asia/Novokuznetsk" {
        value 313;
        description
            "Moscow+03 - Novokuznetsk";
    }
    enum "Asia/Krasnoyarsk" {
        value 314;
        description
            "Moscow+04 - Yenisei River";
    }
    enum "Asia/Irkutsk" {
        value 315;
        description
            "Moscow+05 - Lake Baikal";
    }
    enum "Asia/Yakutsk" {
        value 316;
        description
            "Moscow+06 - Lena River";
    }
    enum "Asia/Vladivostok" {
        value 317;
        description
            "Moscow+07 - Amur River";
    }
    enum "Asia/Sakhalin" {
        value 318;
        description
            "Moscow+07 - Sakhalin Island";
    }
    enum "Asia/Magadan" {
        value 319;
        description
            "Moscow+08 - Magadan";
    }
}
```

```
enum "Asia/Kamchatka" {  
    value 320;  
    description  
        "Moscow+08 - Kamchatka";  
}  
enum "Asia/Anadyr" {  
    value 321;  
    description  
        "Moscow+08 - Bering Sea";  
}  
enum "Africa/Kigali" {  
    value 322;  
}  
enum "Asia/Riyadh" {  
    value 323;  
}  
enum "Pacific/Guadalcanal" {  
    value 324;  
}  
enum "Indian/Mahe" {  
    value 325;  
}  
enum "Africa/Khartoum" {  
    value 326;  
}  
enum "Europe/Stockholm" {  
    value 327;  
}  
enum "Asia/Singapore" {  
    value 328;  
}  
enum "Atlantic/St_Helena" {  
    value 329;  
}  
enum "Europe/Ljubljana" {  
    value 330;  
}  
enum "Arctic/Longyearbyen" {  
    value 331;  
}  
enum "Europe/Bratislava" {  
    value 332;  
}  
enum "Africa/Freetown" {  
    value 333;  
}  
enum "Europe/San_Marino" {  
    value 334;  
}
```

```
}
enum "Africa/Dakar" {
  value 335;
}
enum "Africa/Mogadishu" {
  value 336;
}
enum "America/Paramaribo" {
  value 337;
}
enum "Africa/Juba" {
  value 338;
}
enum "Africa/Sao_Tome" {
  value 339;
}
enum "America/El_Salvador" {
  value 340;
}
enum "America/Lower_Princes" {
  value 341;
}
enum "Asia/Damascus" {
  value 342;
}
enum "Africa/Mbabane" {
  value 343;
}
enum "America/Grand_Turk" {
  value 344;
}
enum "Africa/Ndjamena" {
  value 345;
}
enum "Indian/Kerguelen" {
  value 346;
}
enum "Africa/Lome" {
  value 347;
}
enum "Asia/Bangkok" {
  value 348;
}
enum "Asia/Dushanbe" {
  value 349;
}
enum "Pacific/Fakaofo" {
  value 350;
}
```

```
    }
    enum "Asia/Dili" {
        value 351;
    }
    enum "Asia/Ashgabat" {
        value 352;
    }
    enum "Africa/Tunis" {
        value 353;
    }
    enum "Pacific/Tongatapu" {
        value 354;
    }
    enum "Europe/Istanbul" {
        value 355;
    }
    enum "America/Port_of_Spain" {
        value 356;
    }
    enum "Pacific/Funafuti" {
        value 357;
    }
    enum "Asia/Taipei" {
        value 358;
    }
    enum "Africa/Dar_es_Salaam" {
        value 359;
    }
    enum "Europe/Kiev" {
        value 360;
        description
            "most locations";
    }
    enum "Europe/Uzhgorod" {
        value 361;
        description
            "Ruthenia";
    }
    enum "Europe/Zaporozhye" {
        value 362;
        description
            "Zaporozh'ye, E Lugansk / Zaporizhia, E Luhansk";
    }
    enum "Europe/Simferopol" {
        value 363;
        description
            "central Crimea";
    }
}
```

```
enum "Africa/Kampala" {
    value 364;
}
enum "Pacific/Johnston" {
    value 365;
    description
        "Johnston Atoll";
}
enum "Pacific/Midway" {
    value 366;
    description
        "Midway Islands";
}
enum "Pacific/Wake" {
    value 367;
    description
        "Wake Island";
}
enum "America/New_York" {
    value 368;
    description
        "Eastern Time";
}
enum "America/Detroit" {
    value 369;
    description
        "Eastern Time - Michigan - most locations";
}
enum "America/Kentucky/Louisville" {
    value 370;
    description
        "Eastern Time - Kentucky - Louisville area";
}
enum "America/Kentucky/Monticello" {
    value 371;
    description
        "Eastern Time - Kentucky - Wayne County";
}
enum "America/Indiana/Indianapolis" {
    value 372;
    description
        "Eastern Time - Indiana - most locations";
}
enum "America/Indiana/Vincennes" {
    value 373;
    description
        "Eastern Time - Indiana - Daviess, Dubois, Knox & Martin
        Counties";
}
```



```
}
enum "America/Indiana/Winamac" {
    value 374;
    description
        "Eastern Time - Indiana - Pulaski County";
}
enum "America/Indiana/Marengo" {
    value 375;
    description
        "Eastern Time - Indiana - Crawford County";
}
enum "America/Indiana/Petersburg" {
    value 376;
    description
        "Eastern Time - Indiana - Pike County";
}
enum "America/Indiana/Vevay" {
    value 377;
    description
        "Eastern Time - Indiana - Switzerland County";
}
enum "America/Chicago" {
    value 378;
    description
        "Central Time";
}
enum "America/Indiana/Tell_City" {
    value 379;
    description
        "Central Time - Indiana - Perry County";
}
enum "America/Indiana/Knox" {
    value 380;
    description
        "Central Time - Indiana - Starke County";
}
enum "America/Menominee" {
    value 381;
    description
        "Central Time - Michigan - Dickinson, Gogebic, Iron &
        Menominee Counties";
}
enum "America/North_Dakota/Center" {
    value 382;
    description
        "Central Time - North Dakota - Oliver County";
}
enum "America/North_Dakota/New_Salem" {
```

```
    value 383;
    description
      "Central Time - North Dakota - Morton County (except Mandan
        area)";
  }
  enum "America/North_Dakota/Beulah" {
    value 384;
    description
      "Central Time - North Dakota - Mercer County";
  }
  enum "America/Denver" {
    value 385;
    description
      "Mountain Time";
  }
  enum "America/Boise" {
    value 386;
    description
      "Mountain Time - south Idaho & east Oregon";
  }
  enum "America/Shiprock" {
    value 387;
    description
      "Mountain Time - Navajo";
  }
  enum "America/Phoenix" {
    value 388;
    description
      "Mountain Standard Time - Arizona";
  }
  enum "America/Los_Angeles" {
    value 389;
    description
      "Pacific Time";
  }
  enum "America/Anchorage" {
    value 390;
    description
      "Alaska Time";
  }
  enum "America/Juneau" {
    value 391;
    description
      "Alaska Time - Alaska panhandle";
  }
  enum "America/Sitka" {
    value 392;
    description
```

```
        "Alaska Time - southeast Alaska panhandle";
    }
    enum "America/Yakutat" {
        value 393;
        description
            "Alaska Time - Alaska panhandle neck";
    }
    enum "America/Nome" {
        value 394;
        description
            "Alaska Time - west Alaska";
    }
    enum "America/Adak" {
        value 395;
        description
            "Aleutian Islands";
    }
    enum "America/Metlakatla" {
        value 396;
        description
            "Metlakatla Time - Annette Island";
    }
    enum "Pacific/Honolulu" {
        value 397;
        description
            "Hawaii";
    }
    enum "America/Montevideo" {
        value 398;
    }
    enum "Asia/Samarkand" {
        value 399;
        description
            "west Uzbekistan";
    }
    enum "Asia/Tashkent" {
        value 400;
        description
            "east Uzbekistan";
    }
    enum "Europe/Vatican" {
        value 401;
    }
    enum "America/St_Vincent" {
        value 402;
    }
    enum "America/Caracas" {
        value 403;
```

```
    }
    enum "America/Tortola" {
        value 404;
    }
    enum "America/St_Thomas" {
        value 405;
    }
    enum "Asia/Ho_Chi_Minh" {
        value 406;
    }
    enum "Pacific/Efate" {
        value 407;
    }
    enum "Pacific/Wallis" {
        value 408;
    }
    enum "Pacific/Apia" {
        value 409;
    }
    enum "Asia/Aden" {
        value 410;
    }
    enum "Indian/Mayotte" {
        value 411;
    }
    enum "Africa/Johannesburg" {
        value 412;
    }
    enum "Africa/Lusaka" {
        value 413;
    }
    enum "Africa/Harare" {
        value 414;
    }
  }
}
```

<CODE ENDS>

3. IANA Considerations

This document defines the initial version of the IANA-maintained iana-timezones YANG module.

The iana-timezones module is intended to reflect the IANA "timezone database". When a timezone location is added to the database, the

"iana-timezone" enumeration MUST be updated as defined in RFC 6020 Section 10 to add the newly created timezone location to the enumeration. The new "enum" statement MUST be added to the "iana-timezone" typedef with the same name as the newly added timezone location. A new enum value MUST be allocated by IANA and applied to the newly created enum entry. New entries MAY be placed in any order in the enumeration as long as the previously assigned enumeration values are not changed.

When the iana-timezones YANG module is updated, a new "revision" statement must be added.

This document registers one 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-timezones

Registrant Contact: IANA.

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-timezones

namespace: urn:ietf:params:xml:ns:yang:iana-timezones

prefix: ianatz

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

Jeffrey Lange
GE Digital Energy

Email: jeffrey.k.lange@ge.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2014

M. Bjorklund
Tail-f Systems
July 4, 2013

A YANG Data Model for Interface Management
draft-ietf-netmod-interfaces-cfg-12

Abstract

This document defines a YANG data model for the management of network interfaces. It is expected that interface type specific data models augment the generic interfaces data model defined in this document. The data model includes configuration data, state data and counters for the collection of statistics.

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 5, 2014.

Copyright Notice

Copyright (c) 2013 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
1.2. Tree Diagrams	4
2. Objectives	5
3. Interfaces Data Model	6
3.1. The interface Lists	6
3.2. Interface References	8
3.3. Interface Layering	8
4. Relationship to the IF-MIB	9
5. Interfaces YANG Module	11
6. IANA Considerations	26
7. Security Considerations	27
8. Acknowledgments	28
9. References	29
9.1. Normative References	29
9.2. Informative References	29
Appendix A. Example: Ethernet Interface Module	30
Appendix B. Example: Ethernet Bonding Interface Module	32
Appendix C. Example: VLAN Interface Module	33
Appendix D. Example: NETCONF <get> reply	34
Appendix E. Examples: Interface Naming Schemes	37
E.1. Router with Restricted Interface Names	37
E.2. Router with Arbitrary Interface Names	38
E.3. Ethernet Switch with Restricted Interface Names	39
E.4. Generic Host with Restricted Interface Names	39
E.5. Generic Host with Arbitrary Interface Names	40
Appendix F. ChangeLog	42
F.1. Version -11	42
F.2. Version -08	42
F.3. Version -07	42
F.4. Version -06	42
F.5. Version -05	42
F.6. Version -04	43
F.7. Version -03	43
F.8. Version -02	43
F.9. Version -01	43
Author's Address	44

1. Introduction

This document defines a YANG [RFC6020] data model for the management of network interfaces. It is expected that interface type specific data models augment the generic interfaces data model defined in this document.

Network interfaces are central to the management of many Internet protocols. Thus, it is important to establish a common data model for how interfaces are identified, configured, and monitored.

The data model includes configuration data, state data and counters for the collection of statistics.

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].

The following terms are used within this document:

- o system-controlled interface: An interface is said to be system-controlled if the system creates and deletes the interface independently of what has been explicitly configured. Examples are interfaces representing physical hardware that appear and disappear when hardware (e.g., a line card) is added or removed. System-controlled interfaces may also appear if a certain functionality is enabled (e.g., a loopback interface might appear if the IP protocol stack is enabled).
- o user-controlled interface: An interface is said to be user-controlled if the creation of the interface is controlled by adding explicit interface configuration to the running configuration datastore and the removal of the interface is controlled by removing explicit interface configuration from the running configuration datastore. Examples are VLAN interfaces configured on a system-controlled Ethernet interface.

The following terms are defined in [RFC6241] and are not redefined here:

- o client
- o configuration data

- o server
- o state data

The following terms are defined in [RFC6020] and are not redefined here:

- o augment
- o data model
- o data node

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Objectives

This section describes some of the design objectives for the model presented in Section 5.

- 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 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 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 results from the aggregation of 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.
- o The data model should support both physical interfaces as well as logical interfaces.
- o The data model should include read-only counters in order to gather statistics for octets, packets and errors, sent and received.

3. Interfaces Data Model

This document defines the YANG module "ietf-interfaces", which has the following structure:

```

+--rw interfaces
|   +--rw interface* [name]
|   |   +--rw name                string
|   |   +--rw description?        string
|   |   +--rw type                ianaift:iana-if-type
|   |   +--rw enabled?            boolean
|   |   +--rw link-up-down-trap-enable? enumeration
+--ro interfaces-state
|   +--ro interface* [name]
|   |   +--ro name                string
|   |   +--ro type                ianaift:iana-if-type
|   |   +--ro admin-status        enumeration
|   |   +--ro oper-status         enumeration
|   |   +--ro last-change?        yang:date-and-time
|   |   +--ro if-index            int32
|   |   +--ro phys-address?       yang:phys-address
|   |   +--ro higher-layer-if*    interface-state-ref
|   |   +--ro lower-layer-if*    interface-state-ref
|   |   +--ro speed?              yang:gauge64
|   |   +--ro statistics
|   |   |   +--ro discontinuity-time yang:date-and-time
|   |   |   +--ro in-octets?         yang:counter64
|   |   |   +--ro in-unicast-pkts?   yang:counter64
|   |   |   +--ro in-broadcast-pkts? yang:counter64
|   |   |   +--ro in-multicast-pkts? yang:counter64
|   |   |   +--ro in-discards?      yang:counter32
|   |   |   +--ro in-errors?        yang:counter32
|   |   |   +--ro in-unknown-protos? yang:counter32
|   |   |   +--ro out-octets?       yang:counter64
|   |   |   +--ro out-unicast-pkts?  yang:counter64
|   |   |   +--ro out-broadcast-pkts? yang:counter64
|   |   |   +--ro out-multicast-pkts? yang:counter64
|   |   |   +--ro out-discards?     yang:counter32
|   |   |   +--ro out-errors?       yang:counter32

```

3.1. The interface Lists

The data model for interfaces 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.

There is one list of configured interfaces ("/interfaces/interface"), and a separate list for the operational state of all interfaces

```
("/interfaces-state/interface").
```

It is expected that interface type specific data models augment the interface lists, and possibly 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 {
      ...
    }
  }
}
```

For system-controlled interfaces, the "name" is the device-specific name of the interface. The 'config false' list "/interfaces-state/interface" contains all existing interfaces on the device.

If the device supports arbitrarily named user-controlled interfaces, the NETCONF server advertises the feature "arbitrary-names". If the device does not advertise this feature, the names of user-controlled interfaces MUST match the device's naming scheme. How a client can learn the naming scheme of such devices is outside the scope of this document.

When a system-controlled interface is created by the system, the system tries to apply the interface configuration in /interfaces/interface with the same name as the new interface. If no such interface configuration is found, or if the configured type does not match the real interface type, the system creates the interface without applying explicit configuration.

When a user-controlled interface is created, the configuration determines the name of the interface.

3.2. Interface References

An interface is identified by its name, which is unique within the server. This property is captured in the "interface-ref" and "interface-state-ref" typedefs, which other YANG modules SHOULD use when they need to reference a configured interface or operationally used interface, respectively.

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 data nodes for interface layering, by using "interface-ref" types to reference lower layers.

Below is an example of a model with such nodes. For a more complete example, see Appendix B.

```
import 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";
        }
    }
    // other bonding config params, failover times etc.
}
```

Two state data leaf-lists, "higher-layer-if" and "lower-layer-if", represent a read-only view of the interface layering hierarchy.

4. Relationship to the IF-MIB

If the device implements IF-MIB [RFC2863], each entry in the `/interfaces-state/interface` list is typically mapped to one `ifEntry`. The `if-index` leaf MUST contain the value of the corresponding `ifEntry`'s `ifIndex`.

In most cases, the `name` of an `interface` entry is mapped to `ifName`. `ifName` is defined as an `DisplayString` [RFC2579] which uses a 7-bit ASCII character set. An implementation MUST restrict the allowed values for `name` to match the restrictions of `ifName`.

The IF-MIB allows two different `ifEntries` to have the same `ifName`. Devices that support this feature, and also support the data model defined in this document, cannot have a 1-1 mapping between the `name` leaf and `ifName`.

The configured `description` of an `interface` has traditionally been mapped to `ifAlias` in some implementations. This document allows this mapping, but implementers should be aware of the differences in the value space and persistence for these objects. See the YANG module definition of the leaf `description` in Section 5 for details.

The IF-MIB also defines the writable object `ifPromiscuousMode`. Since this object typically is not a configuration object, it is not mapped to the `ietf-interfaces` module.

There are a number of counters in the IF-MIB that exist in two versions; one with 32 bits and one with 64 bits. The YANG module contains the 64 bits counters only. Note that NETCONF and SNMP may differ in the time granularity in which they provide access to the counters. For example, it is common that SNMP implementations cache counter values for some time.

The following table lists the YANG data nodes with corresponding objects in the IF-MIB.

YANG data node	IF-MIB object
/interfaces-state/interface	ifEntry
/interfaces-state/name	ifName
description	ifAlias
type	ifType
enabled / admin-status	ifAdminStatus
oper-status	ifOperStatus
last-change	ifLastChange
if-index	ifIndex
link-up-down-trap-enable	ifLinkUpDownTrapEnable
phys-address	ifPhysAddress
higher-layer-if / lower-layer-if	ifStackTable
speed	ifSpeed
in-octets	ifHCInOctets
in-unicast-pkts	ifHCInUcastPkts
in-broadcast-pkts	ifHCInBroadcastPkts
in-multicast-pkts	ifHCInMulticastPkts
in-discards	ifInDiscards
in-errors	ifInErrors
in-unknown-protos	ifInUnknownProtos
out-octets	ifHCOctets
out-unicast-pkts	ifHCOctetsUcastPkts
out-broadcast-pkts	ifHCOctetsBroadcastPkts
out-multicast-pkts	ifHCOctetsMulticastPkts
out-discards	ifOutDiscards
out-errors	ifOutErrors

YANG data nodes and related IF-MIB objects

5. Interfaces YANG Module

This YANG module imports typedefs from [I-D.ietf-netmod-rfc6021-bis] and [I-D.ietf-netmod-iana-if-type].

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

<CODE BEGINS> file "ietf-interfaces@2013-07-04.yang"

```
module ietf-interfaces {

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

  import ietf-yang-types {
    prefix yang;
  }
  import iana-if-type {
    prefix ianaift;
  }

  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
    managing network interfaces.

    Copyright (c) 2013 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 2013-07-04 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Interface Management";
}

/* 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
    configured interfaces.";
}

typedef interface-state-ref {
  type leafref {
    path "/if:interfaces-state/if:interface/if:name";
  }
  description
    "This type is used by data models that need to reference
    the operationally present interfaces.";
}

/* Features */

feature arbitrary-names {
  description
    "This feature indicates that the device allows user-controlled
    interfaces to be named arbitrarily.";
}

feature pre-provisioning {
```

```
description
  "This feature indicates that the device supports
  pre-provisioning of interface configuration, i.e., it is
  possible to configure an interface whose physical interface
  hardware is not present on the device.";
}

feature if-mib {
  description
    "This feature indicates that the device implements IF-MIB.";
  reference
    "RFC 2863: The Interfaces Group MIB";
}

/* Data nodes */

container interfaces {
  description
    "Interface configuration parameters.";

  list interface {
    key "name";

    description
      "The list of configured interfaces on the device.

      The operational state of an interface is available in the
      /interfaces-state/interface list. If the configuration of a
      system-controlled interface cannot be used by the system
      (e.g., the interface hardware present does not match the
      interface type), then the configuration is not applied to
      the system-controlled interface shown in the
      /interfaces-state/interface list. If the the configuration
      of a user-controlled interface cannot be used by the system,
      the configured interface is not instantiated in the
      /interfaces-state/interface list.";

    leaf name {
      type string;
      description
        "The name of the interface.

        A device MAY restrict the allowed values for this leaf,
        possibly depending on the type of the interface.

        For system-controlled interfaces, this leaf is the
        device-specific name of the interface. The 'config false'
        list /interfaces-state/interface contains the currently
```

existing interfaces on the device.

If a client tries to create configuration for a system-controlled interface that is not present in the /interfaces-state/interface list, the server MAY reject the request, if the implementation does not support pre-provisioning of interfaces, or if the name refers to an interface that can never exist in the system. A NETCONF server MUST reply with an rpc-error with the error-tag 'invalid-value' in this case.

If the device supports pre-provisioning of interface configuration, the feature 'pre-provisioning' is advertised.

If the device allows arbitrarily named user-controlled interfaces, the feature 'arbitrary-names' is advertised.

When a configured user-controlled interface is created by the system, it is instantiated with the same name in the /interface-state/interface list. Since the name in that list MAY be mapped to ifName by an implementation, such an implementation MUST restrict the allowed values for this leaf so that it matches the restrictions of ifName.

If a NETCONF server that implements this restriction is sent a value that doesn't match the restriction, it MUST reply with an rpc-error with the error-tag 'invalid-value'."

}

```
leaf description {  
  type string;  
  description  
    "A textual description of the interface.
```

This leaf MAY be mapped to ifAlias by an implementation. Such an implementation MUST restrict the allowed values for this leaf so that it matches the restrictions of ifAlias.

If a NETCONF server that implements this restriction is sent a value that doesn't match the restriction, it MUST reply with an rpc-error with the error-tag 'invalid-value'.

Since ifAlias is defined to be stored in non-volatile storage, the MIB implementation MUST map ifAlias to the

value of 'description' in the persistently stored datastore.

Specifically, if the device supports ':startup', when ifAlias is read the device MUST return the value of 'description' in the 'startup' datastore, and when it is written, it MUST be written to the 'running' and 'startup' datastores. Note that it is up to the implementation if it modifies this single leaf in 'startup', or if it performs an implicit copy-config from 'running' to 'startup'.

If the device does not support ':startup', ifAlias MUST be mapped to the 'description' leaf in the 'running' datastore.";

reference

"RFC 2863: The Interfaces Group MIB - ifAlias";

}

leaf type {

type ianaift: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.

If a client tries to set the type of an interface to a value that can never be used by the system, e.g., if the type is not supported or if the type does not match the name of the interface, the server MUST reject the request. A NETCONF server MUST reply with an rpc-error with the error-tag 'invalid-value' in this case.";

reference

"RFC 2863: The Interfaces Group MIB - ifType";

}

leaf enabled {

type boolean;

default "true";

description

"This leaf contains the configured, desired state of the interface.

Systems that implement the IF-MIB use the value of this

leaf in the 'running' datastore to set IF-MIB.ifAdminStatus to 'up' or 'down' after an ifEntry has been initialized, as described in RFC 2863.

Changes in this leaf in the 'running' datastore are reflected in ifAdminStatus, but if ifAdminStatus is changed over SNMP, this leaf is not affected.";

reference

"RFC 2863: The Interfaces Group MIB - ifAdminStatus";

}

leaf link-up-down-trap-enable {

if-feature if-mib;

type enumeration {

enum enabled {

value 1;

}

enum disabled {

value 2;

}

}

description

"Controls whether linkUp/linkDown SNMP notifications 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 (i.e., there are no 'lower-layer-if' entries), and 'disabled' otherwise.";

reference

"RFC 2863: The Interfaces Group MIB -
ifLinkUpDownTrapEnable";

}

}

}

container interfaces-state {

config false;

description

"Data nodes for the operational state of interfaces.";

list interface {

key "name";

description

"The list of interfaces on the device.

System-controlled interfaces created by the system are

```
always present in this list, whether they are configured or
not.";
```

```
leaf name {
  type string;
  description
    "The name of the interface.

    This leaf MAY be mapped to ifName by an implementation.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifName";
}

leaf type {
  type ianaift:iana-if-type;
  mandatory true;
  description
    "The type of the interface.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifType";
}

leaf admin-status {
  if-feature if-mib;
  type enumeration {
    enum up {
      value 1;
      description
        "Ready to pass packets.";
    }
    enum down {
      value 2;
      description
        "Not ready to pass packets and not in some test mode.";
    }
    enum testing {
      value 3;
      description
        "In some test mode.";
    }
  }
  mandatory true;
  description
    "The desired state of the interface.

    This leaf has the same read semantics as ifAdminStatus.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifAdminStatus";
```



```
}  
  
leaf oper-status {  
  type enumeration {  
    enum up {  
      value 1;  
      description  
        "Ready to pass packets.";  
    }  
    enum down {  
      value 2;  
      description  
        "The interface does not pass any packets.";  
    }  
    enum testing {  
      value 3;  
      description  
        "In some test mode.  No operational packets can  
        be passed.";  
    }  
    enum unknown {  
      value 4;  
      description  
        "Status cannot be determined for some reason.";  
    }  
    enum dormant {  
      value 5;  
      description  
        "Waiting for some external event.";  
    }  
    enum not-present {  
      value 6;  
      description  
        "Some component (typically hardware) is missing.";  
    }  
    enum lower-layer-down {  
      value 7;  
      description  
        "Down due to state of lower-layer interface(s).";  
    }  
  }  
  mandatory true;  
  description  
    "The current operational state of the interface.  
  
    This leaf has the same semantics as ifOperStatus.";  
  reference  
    "RFC 2863: The Interfaces Group MIB - ifOperStatus";  
}
```

```
}

leaf last-change {
  type yang:date-and-time;
  description
    "The time the interface entered its current operational
    state.  If the current state was entered prior to the
    last re-initialization of the local network management
    subsystem, then this node is not present.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifLastChange";
}

leaf if-index {
  if-feature if-mib;
  type int32 {
    range "1..2147483647";
  }
  mandatory true;
  description
    "The ifIndex value for the ifEntry represented by this
    interface.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifIndex";
}

leaf phys-address {
  type yang:phys-address;
  description
    "The interface's address at its protocol sub-layer.  For
    example, for an 802.x interface, this object normally
    contains a MAC address.  The interface's media-specific
    modules must define the bit and byte ordering and the
    format of the value of this object.  For interfaces that do
    not have such an address (e.g., a serial line), this node
    is not present.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifPhysAddress";
}

leaf-list higher-layer-if {
  type interface-state-ref;
  description
    "A list of references to interfaces layered on top of this
    interface.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifStackTable";
}
```

```
leaf-list lower-layer-if {
  type interface-state-ref;
  description
    "A list of references to interfaces layered underneath this
    interface.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifStackTable";
}

leaf speed {
  type yang:gauge64;
  units "bits / second";
  description
    "An estimate of the interface's current bandwidth in bits
    per second. For interfaces that do not vary in
    bandwidth or for those where no accurate estimation can
    be made, this node should contain the nominal bandwidth.
    For interfaces that have no concept of bandwidth, this
    node is not present.";
  reference
    "RFC 2863: The Interfaces Group MIB -
    ifSpeed, ifHighSpeed";
}

container statistics {
  description
    "A collection of interface-related statistics objects.";

  leaf discontinuity-time {
    type yang:date-and-time;
    mandatory true;
    description
      "The time on the most recent occasion at which any one or
      more of this interface's counters suffered a
      discontinuity. If no such discontinuities have occurred
      since the last re-initialization of the local management
      subsystem, then this node contains the time the local
      management subsystem re-initialized itself.";
  }

  leaf in-octets {
    type yang:counter64;
    description
      "The total number of octets received on the interface,
      including framing characters.

      Discontinuities in the value of this counter can occur
      at re-initialization of the management system, and at
```

```
        other times as indicated by the value of
        'discontinuity-time'.";
    reference
        "RFC 2863: The Interfaces Group MIB - ifHCInOctets";
}
leaf in-unicast-pkts {
    type yang:counter64;
    description
        "The number of packets, delivered by this sub-layer to a
        higher (sub-)layer, which were not addressed to a
        multicast or broadcast address at this sub-layer.

        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        'discontinuity-time'.";
    reference
        "RFC 2863: The Interfaces Group MIB - ifHCInUcastPkts";
}
leaf in-broadcast-pkts {
    type yang:counter64;
    description
        "The number of packets, delivered by this sub-layer to a
        higher (sub-)layer, which were addressed to a broadcast
        address at this sub-layer.

        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        'discontinuity-time'.";
    reference
        "RFC 2863: The Interfaces Group MIB -
        ifHCInBroadcastPkts";
}
leaf in-multicast-pkts {
    type yang:counter64;
    description
        "The number of packets, delivered by this sub-layer to a
        higher (sub-)layer, which were addressed to a multicast
        address at this sub-layer. For a MAC layer protocol,
        this includes both Group and Functional addresses.

        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        'discontinuity-time'.";
    reference
        "RFC 2863: The Interfaces Group MIB -
```

```
        ifHCInMulticastPkts";
    }
    leaf in-discards {
        type yang:counter32;
        description
            "The number of inbound packets which were chosen to be
            discarded even though no errors had been detected to
            prevent their being deliverable to a higher-layer
            protocol. One possible reason for discarding such a
            packet could be to free up buffer space.

            Discontinuities in the value of this counter can occur
            at re-initialization of the management system, and at
            other times as indicated by the value of
            'discontinuity-time'.";
        reference
            "RFC 2863: The Interfaces Group MIB - ifInDiscards";
    }
    leaf in-errors {
        type yang:counter32;
        description
            "For packet-oriented interfaces, the number of inbound
            packets that contained errors preventing them from being
            deliverable to a higher-layer protocol. For character-
            oriented or fixed-length interfaces, the number of
            inbound transmission units that contained errors
            preventing them from being deliverable to a higher-layer
            protocol.

            Discontinuities in the value of this counter can occur
            at re-initialization of the management system, and at
            other times as indicated by the value of
            'discontinuity-time'.";
        reference
            "RFC 2863: The Interfaces Group MIB - ifInErrors";
    }
    leaf in-unknown-protos {
        type yang:counter32;
        description
            "For packet-oriented interfaces, the number of packets
            received via the interface which were discarded because
            of an unknown or unsupported protocol. For
            character-oriented or fixed-length interfaces that
            support protocol multiplexing the number of transmission
            units received via the interface which were discarded
            because of an unknown or unsupported protocol. For any
            interface that does not support protocol multiplexing,
            this counter is not present.
```

```
        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        'discontinuity-time'.";
    reference
        "RFC 2863: The Interfaces Group MIB - ifInUnknownProtos";
}

leaf out-octets {
    type yang:counter64;
    description
        "The total number of octets transmitted out of the
        interface, including framing characters.

        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        'discontinuity-time'.";
    reference
        "RFC 2863: The Interfaces Group MIB - ifHCOutOctets";
}

leaf out-unicast-pkts {
    type yang:counter64;
    description
        "The total number of packets that higher-level protocols
        requested be transmitted, and which were not addressed
        to a multicast or broadcast address at this sub-layer,
        including those that were discarded or not sent.

        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        'discontinuity-time'.";
    reference
        "RFC 2863: The Interfaces Group MIB - ifHCOutUcastPkts";
}

leaf out-broadcast-pkts {
    type yang:counter64;
    description
        "The total number of packets that higher-level protocols
        requested be transmitted, and which were addressed to a
        broadcast address at this sub-layer, including those
        that were discarded or not sent.

        Discontinuities in the value of this counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        'discontinuity-time'.";
```

```
reference
  "RFC 2863: The Interfaces Group MIB -
    ifHCOutBroadcastPkts";
}
leaf out-multicast-pkts {
  type yang:counter64;
  description
    "The total number of packets that higher-level protocols
    requested be transmitted, and which were addressed to a
    multicast address at this sub-layer, including those
    that were discarded or not sent. For a MAC layer
    protocol, this includes both Group and Functional
    addresses.

    Discontinuities in the value of this counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    'discontinuity-time'.";
  reference
    "RFC 2863: The Interfaces Group MIB -
      ifHCOutMulticastPkts";
}
leaf out-discards {
  type yang:counter32;
  description
    "The number of outbound packets which were chosen to be
    discarded even though no errors had been detected to
    prevent their being transmitted. One possible reason
    for discarding such a packet could be to free up buffer
    space.

    Discontinuities in the value of this counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    'discontinuity-time'.";
  reference
    "RFC 2863: The Interfaces Group MIB - ifOutDiscards";
}
leaf out-errors {
  type yang:counter32;
  description
    "For packet-oriented interfaces, the number of outbound
    packets that could not be transmitted because of errors.
    For character-oriented or fixed-length interfaces, the
    number of outbound transmission units that could not be
    transmitted because of errors.

    Discontinuities in the value of this counter can occur
```

```
        at re-initialization of the management system, and at
        other times as indicated by the value of
        'discontinuity-time'.";
    reference
        "RFC 2863: The Interfaces Group MIB - ifOutErrors";
    }
}
}
```

<CODE ENDS>

6. 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 IESG.

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

7. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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:

/interfaces/interface: This list specifies the configured interfaces on a device. Unauthorized access to this list could cause the device to ignore packets it should receive and process.

/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 it should receive and process.

8. Acknowledgments

The author wishes to thank Alexander Clemm, Per Hedeland, Ladislav Lhotka, and Juergen Schoenwaelder for their helpful comments.

9. References

9.1. Normative References

- [I-D.ietf-netmod-iana-if-type]
Bjorklund, M., "IANA Interface Type YANG Module",
draft-ietf-netmod-iana-if-type-07 (work in progress),
July 2013.
- [I-D.ietf-netmod-rfc6021-bis]
Schoenwaelder, J., "Common YANG Data Types",
draft-ietf-netmod-rfc6021-bis-03 (work in progress),
June 2013.
- [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.

9.2. Informative References

- [RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J.
Schoenwaelder, Ed., "Textual Conventions for SMIV2",
STD 58, RFC 2579, April 1999.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A.
Bierman, "Network Configuration Protocol (NETCONF)",
RFC 6241, June 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure
Shell (SSH)", RFC 6242, June 2011.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration
Protocol (NETCONF) Access Control Model", RFC 6536,
March 2012.

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 also shows how operational state parameters can be conditionally augmented to the operational interface list. The example 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;
  }

  // configuration parameters for ethernet interfaces
  augment "/if:interfaces/if:interface" {
    when "if:type = 'ethernetCsmacd'";

    container ethernet {
      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...
}
```

```
    }

    // operational state parameters for ethernet interfaces
    augment "/if:interfaces-state/if:interface" {
        when "if:type = 'ethernetCsmacd'";

        container ethernet {
            leaf duplex {
                type enumeration {
                    enum "half";
                    enum "full";
                }
            }
            // 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" {
        description
          "If a vlan-id is defined, a base-interface must
           be specified.";
      }
    }
  }
}
```


Appendix D. Example: NETCONF <get> 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"
      xmlns:vlan="http://example.com/vlan">

      <interface>
        <name>eth0</name>
        <type>ethernetCsmacd</type>
        <enabled>false</enabled>
      </interface>

      <interface>
        <name>eth1</name>
        <type>ethernetCsmacd</type>
        <enabled>true</enabled>
        <vlan:vlan-tagging>true</vlan:vlan-tagging>
      </interface>

      <interface>
        <name>eth1.10</name>
        <type>l2vlan</type>
        <enabled>true</enabled>
        <vlan:base-interface>eth1</vlan:base-interface>
        <vlan:vlan-id>10</vlan:vlan-id>
      </interface>

      <interface>
        <name>lo1</name>
        <type>softwareLoopback</type>
        <enabled>true</enabled>
      </interface>

    </interfaces>

    <interfaces-state
      xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">

      <interface>
        <name>eth0</name>
```

```
<type>ethernetCsmacd</type>
<admin-status>down</admin-status>
<oper-status>down</oper-status>
<if-index>2</if-index>
<phys-address>00:01:02:03:04:05</phys-address>
<statistics>
  <discontinuity-time>
    2013-04-01T03:00:00+00:00
  </discontinuity-time>
  <!-- counters now shown here -->
</statistics>
</interface>

<interface>
  <name>eth1</name>
  <type>ethernetCsmacd</type>
  <admin-status>up</admin-status>
  <oper-status>up</oper-status>
  <if-index>7</if-index>
  <phys-address>00:01:02:03:04:06</phys-address>
  <higher-layer-if>eth1.10</higher-layer-if>
  <statistics>
    <discontinuity-time>
      2013-04-01T03:00:00+00:00
    </discontinuity-time>
    <!-- counters now shown here -->
  </statistics>
</interface>

<interface>
  <name>eth1.10</name>
  <type>l2vlan</type>
  <admin-status>up</admin-status>
  <oper-status>up</oper-status>
  <if-index>9</if-index>
  <lower-layer-if>eth1</lower-layer-if>
  <statistics>
    <discontinuity-time>
      2013-04-01T03:00:00+00:00
    </discontinuity-time>
    <!-- counters now shown here -->
  </statistics>
</interface>

<!-- This interface is not configured -->
<interface>
  <name>eth2</name>
  <type>ethernetCsmacd</type>
```

```
<admin-status>down</admin-status>
<oper-status>down</oper-status>
<if-index>8</if-index>
<phys-address>00:01:02:03:04:07</phys-address>
<statistics>
  <discontinuity-time>
    2013-04-01T03:00:00+00:00
  </discontinuity-time>
  <!-- counters now shown here -->
</statistics>
</interface>

<interface>
  <name>lo1</name>
  <type>softwareLoopback</type>
  <admin-status>up</admin-status>
  <oper-status>up</oper-status>
  <if-index>1</if-index>
  <statistics>
    <discontinuity-time>
      2013-04-01T03:00:00+00:00
    </discontinuity-time>
    <!-- counters now shown here -->
  </statistics>
</interface>

</interfaces-state>
</data>
</rpc-reply>
```

Appendix E. Examples: Interface Naming Schemes

This section gives examples of some implementation strategies.

The examples make use of the example data model "ex-vlan" (see Appendix C) to show how user-controlled interfaces can be configured.

E.1. Router with Restricted Interface Names

In this example, a router has support for 4 line cards, each with 8 ports. The slots for the cards are physically numbered from 0 to 3, and the ports on each card from 0 to 7. Each card has fast- or gigabit-ethernet ports.

The device-specific names for these physical interfaces are "fastethernet-N/M" or "gigabitethernet-N/M".

The name of a vlan interface is restricted to the form "<physical-interface-name>.<subinterface-number>".

It is assumed that the operator is aware of this naming scheme. The implementation auto-initializes the value for "type" based on the interface name.

The NETCONF server does not advertise the 'arbitrary-names' feature in the <hello> message.

An operator can configure a physical interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>fastethernet-1/0</name>
</interface>
```

When the server processes this request, it will set the leaf "type" to "ethernetCsmacd". Thus, if the client performs a <get-config> right after the <edit-config> above, it will get:

```
<interface>
  <name>fastethernet-1/0</name>
  <type>ethernetCsmacd</type>
</interface>
```

The client can configure a vlan interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>fastethernet-1/0.10005</name>
  <type>l2-vlan</type>
  <vlan:base-interface>fastethernet-1/0</vlan:base-interface>
  <vlan:vlan-id>5</vlan:vlan-id>
</interface>
```

If the client tries to change the type of the physical interface with an `<edit-config>` containing:

```
<interface nc:operation="merge">
  <name>fastethernet-1/0</name>
  <type>tunnel</type>
</interface>
```

then the server will reply with an "invalid-value" error, since the new type does not match the name.

E.2. Router with Arbitrary Interface Names

In this example, a router has support for 4 line cards, each with 8 ports. The slots for the cards are physically numbered from 0 to 3, and the ports on each card from 0 to 7. Each card has fast- or gigabit-ethernet ports.

The device-specific names for these physical interfaces are "fastethernet-N/M" or "gigabitethernet-N/M".

The implementation does not restrict the user-controlled interface names. This allows to more easily apply the interface configuration to a different interface. However, the additional level of indirection also makes it a bit more complex to map interface names found in other protocols to configuration entries.

The NETCONF server advertises the 'arbitrary-names' feature in the `<hello>` message.

Physical interfaces are configured as in Appendix E.1.

An operator can configure a VLAN interface by sending an `<edit-config>` containing:

```
<interface nc:operation="create">
  <name>acme-interface</name>
  <type>l2-vlan</type>
  <vlan:base-interface>fastethernet-1/0</vlan:base-interface>
  <vlan:vlan-id>5</vlan:vlan-id>
</interface>
```

If necessary, the operator can move the configuration named "acme-interface" over to a different physical interface with an `<edit-config>` containing:

```
<interface nc:operation="merge">
  <name>acme-interface</name>
  <vlan:base-interface>fastethernet-1/1</vlan:base-interface>
</interface>
```

E.3. Ethernet Switch with Restricted Interface Names

In this example, an ethernet switch has a number of ports, each port identified by a simple port number.

The device-specific names for the physical interfaces are numbers that match the physical port number.

An operator can configure a physical interface by sending an `<edit-config>` containing:

```
<interface nc:operation="create">
  <name>6</name>
</interface>
```

When the server processes this request, it will set the leaf "type" to "ethernetCsmacd". Thus, if the client performs a `<get-config>` right after the `<edit-config>` above, it will get:

```
<interface>
  <name>6</name>
  <type>ethernetCsmacd</type>
</interface>
```

E.4. Generic Host with Restricted Interface Names

In this example, a generic host has interfaces named by the kernel. The system identifies the physical interface by the name assigned by the operating system to the interface.

The name of a vlan interface is restricted to the form "`<physical-interface-name>:<vlan-number>`".

The NETCONF server does not advertise the 'arbitrary-names' feature in the `<hello>` message.

An operator can configure an interface by sending an `<edit-config>` containing:

```
<interface nc:operation="create">
  <name>eth8</name>
</interface>
```

When the server processes this request, it will set the leaf "type" to "ethernetCsmacd". Thus, if the client performs a <get-config> right after the <edit-config> above, it will get:

```
<interface>
  <name>eth8</name>
  <type>ethernetCsmacd</type>
</interface>
```

The client can configure a vlan interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>eth8:5</name>
  <type>l2-vlan</type>
  <vlan:base-interface>eth8</vlan:base-interface>
  <vlan:vlan-id>5</vlan:vlan-id>
</interface>
```

E.5. Generic Host with Arbitrary Interface Names

In this example, a generic host has interfaces named by the kernel. The system identifies the physical interface by the name assigned by the operating system to the interface.

The implementation does not restrict the user-controlled interface names. This allows to more easily apply the interface configuration to a different interface. However, the additional level of indirection also makes it a bit more complex to map interface names found in other protocols to configuration entries.

The NETCONF server advertises the 'arbitrary-names' feature in the <hello> message.

Physical interfaces are configured as in Appendix E.4.

An operator can configure a VLAN interface by sending an <edit-config> containing:

```
<interface nc:operation="create">
  <name>acme-interface</name>
  <type>l2-vlan</type>
  <vlan:base-interface>eth8</vlan:base-interface>
  <vlan:vlan-id>5</vlan:vlan-id>
```

```
</interface>
```

If necessary, the operator can move the configuration named "acme-interface" over to a different physical interface with an <edit-config> containing:

```
<interface nc:operation="merge">
  <name>acme-interface</name>
  <vlan:base-interface>eth3</vlan:base-interface>
</interface>
```


Appendix F. ChangeLog

RFC Editor: remove this section upon publication as an RFC.

F.1. Version -11

- o Separated the operational state from the configuration.
- o Removed 'location', and instead use the name to identify physical interfaces.
- o Added the feature 'pre-provisioning'.
- o Made 'oper-status' and 'if-index' mandatory in the data model.
- o Added 'admin-status'.
- o Clarified why description can be mapped to ifAlias.
- o Clarified that 64-bit counters only are used, where there exist 64-bit and 32-bit counters in IF-MIB.
- o Updated Security Considerations section with a reference to NACM.

F.2. Version -08

- o Removed the mtu leaf.
- o Added examples of different interface naming schemes.

F.3. Version -07

- o Made leaf speed config false.

F.4. Version -06

- o Added oper-status leaf.
- o Added leaf-lists higher-layer-if and lower-layer-if, that show the interface layering.
- o Added container statistics with counters.

F.5. Version -05

- o Added an Informative References section.

- o Updated the Security Considerations section.
- o Clarified the behavior of an NETCONF server when invalid values are received.

F.6. Version -04

- o Clarified why ifPromiscuousMode is not part of this data model.
- o Added a table that shows the mapping between this YANG data model and IF-MIB.

F.7. Version -03

- o Added the section Relationship to the IF-MIB.
- o Changed if-index to be a leaf instead of leaf-list.
- o Explained the notation used in the data model tree picture.

F.8. Version -02

- o Editorial fixes

F.9. 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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2014

M. Bjorklund
Tail-f Systems
October 18, 2013

A YANG Data Model for IP Management
draft-ietf-netmod-ip-cfg-11

Abstract

This document defines a YANG data model for management of IP implementations.

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 21, 2014.

Copyright Notice

Copyright (c) 2013 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
1.2. Tree Diagrams	3
2. IP Data Model	5
3. Relationship to IP-MIB	7
4. IP management YANG Module	9
5. IANA Considerations	24
6. Security Considerations	25
7. Acknowledgments	27
8. References	28
8.1. Normative References	28
8.2. Informative References	28
Appendix A. Example: NETCONF <get> reply	30
Author's Address	32

1. Introduction

This document defines a YANG [RFC6020] data model for management of IP implementations.

The data model covers configuration of per-interface IPv4 and IPv6 parameters, and mappings of IP addresses to link-layer addresses. It also provides information about which IP addresses are operationally used, and which link-layer mappings exist.

Parameters to manage IP routing are defined in [I-D.ietf-netmod-routing-cfg].

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].

The following terms are defined in [RFC6241] and are not redefined here:

- o client
- o server

The following terms are defined in [RFC6020] and are not redefined here:

- o augment
- o data model
- o data node

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).

- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2. IP Data Model

This document defines the YANG module "ietf-ip", which augments the "interface" and "interface-state" lists defined in the "ietf-interfaces" module [I-D.ietf-netmod-interfaces-cfg] with IP specific nodes, and adds IP specific state data.

The data model has the following structure for IP configuration per interface:

```

+--rw if:interfaces
  +--rw if:interface* [name]
    ...
    +--rw ipv4!
      +--rw enabled?          boolean
      +--rw forwarding?       boolean
      +--rw mtu?              uint16
      +--rw address* [ip]
        +--rw ip              inet:ipv4-address-no-zone
        +--rw (subnet)
          +--:(prefix-length)
            +--rw ip:prefix-length?  uint8
          +--:(netmask)
            +--rw ip:netmask?        yang:dotted-quad
      +--rw neighbor* [ip]
        +--rw ip              inet:ipv4-address-no-zone
        +--rw link-layer-address  yang:phys-address
    +--rw ipv6!
      +--rw enabled?          boolean
      +--rw forwarding?       boolean
      +--rw mtu?              uint32
      +--rw address* [ip]
        +--rw ip              inet:ipv6-address-no-zone
        +--rw prefix-length    uint8
      +--rw neighbor* [ip]
        +--rw ip              inet:ipv6-address-no-zone
        +--rw link-layer-address  yang:phys-address
      +--rw dup-addr-detect-transmits?  uint32
      +--rw autoconf
        +--rw create-global-addresses?    boolean
        +--rw create-temporary-addresses?  boolean
        +--rw temporary-valid-lifetime?    uint32
        +--rw temporary-preferred-lifetime? uint32

```

The data model defines two configuration containers per interface, "ipv4" and "ipv6", representing the IPv4 and IPv6 address families. In each container, there is a leaf "enabled" that controls if the address family is enabled on that interface, and a leaf "forwarding"

that controls if IP packet forwarding for the address family is enabled on the interface. In each container, there is also a list of configured addresses, and a list of configured mappings from IP addresses to link-layer addresses.

The data model has the following structure for IP state per interface:

```

+--ro if:interfaces-state
  +--ro if:interface* [name]
    ...
    +--ro ipv4!
      +--ro forwarding?    boolean
      +--ro mtu?           uint16
      +--ro address* [ip]
        +--ro ip           inet:ipv4-address-no-zone
        +--ro (subnet)?
          +--:(prefix-length)
            +--ro prefix-length?    uint8
          +--:(netmask)
            +--ro netmask?          yang:dotted-quad
        +--ro origin?          ip-address-origin
      +--ro neighbor* [ip]
        +--ro ip             inet:ipv4-address-no-zone
        +--ro link-layer-address? yang:phys-address
        +--ro origin?        neighbor-origin
    +--ro ipv6!
      +--ro forwarding?    boolean
      +--ro mtu?           uint32
      +--ro address* [ip]
        +--ro ip           inet:ipv6-address-no-zone
        +--ro prefix-length    uint8
        +--ro origin?         ip-address-origin
        +--ro status?         enumeration
      +--ro neighbor* [ip]
        +--ro ip             inet:ipv6-address-no-zone
        +--ro link-layer-address? yang:phys-address
        +--ro origin?        neighbor-origin
        +--ro is-router?     empty
        +--ro state?         enumeration

```

The data model defines two state containers per interface, "ipv4" and "ipv6", representing the IPv4 and IPv6 address families. In each container, there is a leaf "forwarding" that indicates if IP packet forwarding is enabled on that interface. In each container there is also a list of all addresses in use, and a list of known mappings from IP addresses to link-layer addresses.

3. Relationship to IP-MIB

If the device implements IP-MIB [RFC4293], each entry in the "ipv4/address" and "ipv6/address" lists is mapped to one `ipAddressEntry`, where the `ipAddressIfIndex` refers to the "address" entry's interface.

The IP-MIB defines objects to control IPv6 Router Advertisement. The corresponding YANG data nodes are defined in [I-D.ietf-netmod-routing-cfg].

The entries in "ipv4/neighbor" and "ipv6/neighbor" are mapped to `ipNetToPhysicalTable`.

The following tables list the YANG data nodes with corresponding objects in the IP-MIB.

YANG data node in /if:interfaces/if:interface	IP-MIB object
ipv4/enabled	<code>ipv4InterfaceEnableStatus</code>
ipv4/address	<code>ipAddressEntry</code>
ipv4/address/ip	<code>ipAddressAddrType</code> <code>ipAddressAddr</code>
ipv4/neighbor	<code>ipNetToPhysicalEntry</code>
ipv4/neighbor/ip	<code>ipNetToPhysicalNetAddressType</code> <code>ipNetToPhysicalNetAddressAddr</code>
ipv4/neighbor/link-layer-address	<code>ipNetToPhysicalPhysAddress</code>
ipv6/enabled	<code>ipv6InterfaceEnableStatus</code>
ipv6/forwarding	<code>ipv6InterfaceForwarding</code>
ipv6/address	<code>ipAddressEntry</code>
ipv6/address/ip	<code>ipAddressAddrType</code> <code>ipAddressAddr</code>
ipv6/neighbor	<code>ipNetToPhysicalEntry</code>
ipv6/neighbor/link-layer-address	<code>ipNetToPhysicalPhysAddress</code>
ipv6/neighbor/origin	<code>ipNetToPhysicalType</code>

YANG interface configuration data nodes and related IP-MIB objects

YANG data node in /if:interfaces-state/if:interface	IP-MIB object
ipv4	ipv4InterfaceEnableStatus
ipv4/address	ipAddressEntry
ipv4/address/ip	ipAddressAddrType
	ipAddressAddr
ipv4/address/origin	ipAddressOrigin
ipv4/neighbor	ipNetToPhysicalEntry
ipv4/neighbor/interface	ipNetToPhysicalIfIndex
ipv4/neighbor/ip	ipNetToPhysicalNetAddressType
	ipNetToPhysicalNetAddressAddr
ipv4/neighbor/link-layer-address	ipNetToPhysicalPhysAddress
ipv4/neighbor/origin	ipNetToPhysicalType
ipv6	ipv6InterfaceEnableStatus
ipv6/forwarding	ipv6InterfaceForwarding
ipv6/address	ipAddressEntry
ipv6/address/ip	ipAddressAddrType
	ipAddressAddr
ipv6/address/origin	ipAddressOrigin
ipv6/address/status	ipAddressStatus
ipv6/neighbor	ipNetToPhysicalEntry
ipv6/neighbor/interface	ipNetToPhysicalIfIndex
ipv6/neighbor/ip	ipNetToPhysicalNetAddressType
	ipNetToPhysicalNetAddressAddr
ipv6/neighbor/link-layer-address	ipNetToPhysicalPhysAddress
ipv6/neighbor/origin	ipNetToPhysicalType
ipv6/neighbor/state	ipNetToPhysicalState

YANG interface state data nodes and related IP-MIB objects

4. IP management YANG Module

This module imports typedefs from [RFC6991] and [I-D.ietf-netmod-interfaces-cfg], and references [RFC0791], [RFC0826], [RFC2460], [RFC4861], [RFC4862], and [RFC4941].

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

<CODE BEGINS> file "ietf-ip@2013-10-18.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;  
    }  
    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:   Martin Bjorklund  
                  <mailto:mbj@tail-f.com>";  
  
    description  
        "This module contains a collection of YANG definitions for  
        configuring IP implementations.  
  
        Copyright (c) 2013 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 2013-10-18 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for IP Management";
}

/*
 * Features
 */

feature ipv4-non-contiguous-netmasks {
  description
    "Indicates support for configuring non-contiguous
    subnet masks.";
}

feature ipv6-privacy-autoconf {
  description
    "Indicates support for Privacy Extensions for Stateless Address
    Autoconfiguration in IPv6.";
  reference
    "RFC 4941: Privacy Extensions for Stateless Address
    Autoconfiguration in IPv6";
}

/*
 * Typedefs
 */

typedef ip-address-origin {
  type enumeration {
    enum other {
```

```
        description
            "None of the following.";
    }
    enum static {
        description
            "Indicates that the address has been statically
            configured, for example using NETCONF or a Command Line
            Interface.";
    }
    enum dhcp {
        description
            "Indicates an address that has been assigned to this
            system by a DHCP server.";
    }
    enum link-layer {
        description
            "Indicates an address created by IPv6 stateless
            auto-configuration.";
    }
    enum random {
        description
            "Indicates an address chosen by the system at
            random, e.g., an IPv4 address within 169.254/16, or an
            RFC 4941 privacy address.";
    }
}
description
    "The origin of an address.";
}

typedef neighbor-origin {
    type enumeration {
        enum other {
            description
                "None of the following.";
        }
        enum static {
            description
                "Indicates that the mapping has been statically
                configured, for example using NETCONF or a Command Line
                Interface.";
        }
        enum dynamic {
            description
                "Indicates that the mapping has been dynamically resolved
                using e.g., IPv4 ARP or the IPv6 Neighbor Discovery
                protocol.";
        }
    }
}
```

```
    }
    description
      "The origin of a neighbor entry.";
  }

  /*
   * Configuration data nodes
   */

  augment "/if:interfaces/if:interface" {
    description
      "Parameters for configuring IP on interfaces.

      If an interface is not capable of running IP, the server
      must not allow the client to configure these parameters.";

    container ipv4 {
      presence
        "Enables IPv4 unless the 'enabled' leaf
        (which defaults to 'true') is set to 'false'";
      description
        "Parameters for the IPv4 address family.";

      leaf enabled {
        type boolean;
        default true;
        description
          "Controls if IPv4 is enabled or disabled on this
          interface.";
      }

      leaf forwarding {
        type boolean;
        default false;
        description
          "Controls if IPv4 packet forwarding is enabled or disabled
          on this interface.";
      }

      leaf mtu {
        type uint16 {
          range "68..max";
        }
        units octets;
        description
          "The size, in octets, of the largest IPv4 packet that the
          interface will send and receive.

          The server may restrict the allowed values for this leaf
          depending on the interface's type."
      }
    }
  }
}
```



```
        If this leaf is not configured, the operationally used mtu
        depends on the interface's type.";
    reference
        "RFC 791: Internet Protocol";
}
list address {
    key "ip";
    description
        "The list of configured IPv4 addresses on the interface.";

    leaf ip {
        type inet:ipv4-address-no-zone;
        description
            "The IPv4 address on the interface.";
    }
    choice subnet {
        mandatory true;
        description
            "The subnet can be specified as a prefix-length, or,
            if the server supports non-contiguous netmasks, as
            a netmask.";
        leaf prefix-length {
            type uint8 {
                range "0..32";
            }
            description
                "The length of the subnet prefix.";
        }
        leaf netmask {
            if-feature ipv4-non-contiguous-netmasks;
            type yang:dotted-quad;
            description
                "The subnet specified as a netmask.";
        }
    }
}
list neighbor {
    key "ip";
    description
        "A list of mappings from IPv4 addresses to
        link-layer addresses.

        Entries in this list are used as static entries in the
        ARP cache.";
    reference
        "RFC 826: An Ethernet Address Resolution Protocol";

    leaf ip {
```

```
        type inet:ipv4-address-no-zone;
        description
            "The IPv4 address of the neighbor node.";
    }
    leaf link-layer-address {
        type yang:phys-address;
        mandatory true;
        description
            "The link-layer address of the neighbor node.";
    }
}

container ipv6 {
    presence
        "Enables IPv6 unless the 'enabled' leaf
        (which defaults to 'true') is set to 'false'";
    description
        "Parameters for the IPv6 address family.";

    leaf enabled {
        type boolean;
        default true;
        description
            "Controls if IPv6 is enabled or disabled on this
            interface.";
    }

    leaf forwarding {
        type boolean;
        default false;
        description
            "Controls if IPv6 packet forwarding is enabled or disabled
            on this interface.";
        reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)
            Section 6.2.1, IsRouter";
    }

    leaf mtu {
        type uint32 {
            range "1280..max";
        }
        units octets;
        description
            "The size, in octets, of the largest IPv6 packet that the
            interface will send and receive.

            The server may restrict the allowed values for this leaf
            depending on the interface's type."
    }
}
```

```
        If this leaf is not configured, the operationally used mtu
        depends on the interface's type.";
    reference
        "RFC 2460: IPv6 Specification
         Section 5";
}
list address {
    key "ip";
    description
        "The list of configured IPv6 addresses on the interface.";

    leaf ip {
        type inet:ipv6-address-no-zone;
        description
            "The IPv6 address on the interface.";
    }
    leaf prefix-length {
        type uint8 {
            range "0..128";
        }
        mandatory true;
        description
            "The length of the subnet prefix.";
    }
}
list neighbor {
    key "ip";
    description
        "A list of mappings from IPv6 addresses to
        link-layer addresses.

        Entries in this list are used as static entries in the
        Neighbor Cache.";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)";

    leaf ip {
        type inet:ipv6-address-no-zone;
        description
            "The IPv6 address of the neighbor node.";
    }
    leaf link-layer-address {
        type yang:phys-address;
        mandatory true;
        description
            "The link-layer address of the neighbor node.";
    }
}
```

```
leaf dup-addr-detect-transmits {
  type uint32;
  default 1;
  description
    "The number of consecutive Neighbor Solicitation messages
     sent while performing Duplicate Address Detection on a
     tentative address. A value of zero indicates that
     Duplicate Address Detection is not performed on
     tentative addresses. A value of one indicates a single
     transmission with no follow-up retransmissions.";
  reference
    "RFC 4862: IPv6 Stateless Address Autoconfiguration";
}
container autoconf {
  description
    "Parameters to control the autoconfiguration of IPv6
     addresses, as described in RFC 4862.";
  reference
    "RFC 4862: IPv6 Stateless Address Autoconfiguration";

  leaf create-global-addresses {
    type boolean;
    default true;
    description
      "If enabled, the host creates global addresses as
       described in section 5.5 of RFC 4862.";
    reference
      "RFC 4862: IPv6 Stateless Address Autoconfiguration";
  }
  leaf create-temporary-addresses {
    if-feature ipv6-privacy-autoconf;
    type boolean;
    default false;
    description
      "If enabled, the host creates temporary addresses as
       described in RFC 4941.";
    reference
      "RFC 4941: Privacy Extensions for Stateless Address
       Autoconfiguration in IPv6";
  }
  leaf temporary-valid-lifetime {
    if-feature ipv6-privacy-autoconf;
    type uint32;
    units "seconds";
    default 604800;
    description
      "The time period during which the temporary address
       is valid.";
```

```
        reference
        "RFC 4941: Privacy Extensions for Stateless Address
        Autoconfiguration in IPv6
        - TEMP_VALID_LIFETIME";
    }
    leaf temporary-preferred-lifetime {
        if-feature ipv6-privacy-autoconf;
        type uint32;
        units "seconds";
        default 86400;
        description
            "The time period during which the temporary address is
            preferred.";
        reference
        "RFC 4941: Privacy Extensions for Stateless Address
        Autoconfiguration in IPv6
        - TEMP_PREFERRED_LIFETIME";
    }
}
}
}

/*
 * Operational state data nodes
 */

augment "/if:interfaces-state/if:interface" {
    description
        "Data nodes for the operational state of IP on interfaces.";

    container ipv4 {
        presence "Present if IPv4 is enabled on this interface";
        config false;
        description
            "Interface specific parameters for the IPv4 address family.";

        leaf forwarding {
            type boolean;
            description
                "Indicates if IPv4 packet forwarding is enabled or disabled
                on this interface.";
        }
        leaf mtu {
            type uint16 {
                range "68..max";
            }
            units octets;
            description

```

```
        "The size, in octets, of the largest IPv4 packet that the
        interface will send and receive.";
    reference
        "RFC 791: Internet Protocol";
}
list address {
    key "ip";
    description
        "The list of IPv4 addresses on the interface.";

    leaf ip {
        type inet:ipv4-address-no-zone;
        description
            "The IPv4 address on the interface.";
    }
    choice subnet {
        description
            "The subnet can be specified as a prefix-length, or,
            if the server supports non-contiguous netmasks, as
            a netmask.";
        leaf prefix-length {
            type uint8 {
                range "0..32";
            }
            description
                "The length of the subnet prefix.";
        }
        leaf netmask {
            if-feature ipv4-non-contiguous-netmasks;
            type yang:dotted-quad;
            description
                "The subnet specified as a netmask.";
        }
    }
}
leaf origin {
    type ip-address-origin;
    description
        "The origin of this address.";
}
}
list neighbor {
    key "ip";
    description
        "A list of mappings from IPv4 addresses to
        link-layer addresses.

        This list represents the ARP Cache.";
    reference
```

```
    "RFC 826: An Ethernet Address Resolution Protocol";

    leaf ip {
        type inet:ipv4-address-no-zone;
        description
            "The IPv4 address of the neighbor node.";
    }
    leaf link-layer-address {
        type yang:phys-address;
        description
            "The link-layer address of the neighbor node.";
    }
    leaf origin {
        type neighbor-origin;
        description
            "The origin of this neighbor entry.";
    }
}

container ipv6 {
    presence "Present if IPv6 is enabled on this interface";
    config false;
    description
        "Parameters for the IPv6 address family.";

    leaf forwarding {
        type boolean;
        default false;
        description
            "Indicates if IPv6 packet forwarding is enabled or disabled
             on this interface.";
        reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)
             Section 6.2.1, IsRouter";
    }
    leaf mtu {
        type uint32 {
            range "1280..max";
        }
        units octets;
        description
            "The size, in octets, of the largest IPv6 packet that the
             interface will send and receive.";
        reference
            "RFC 2460: IPv6 Specification
             Section 5";
    }
}
```

```
}
list address {
  key "ip";
  description
    "The list of IPv6 addresses on the interface.";

  leaf ip {
    type inet:ipv6-address-no-zone;
    description
      "The IPv6 address on the interface.";
  }
  leaf prefix-length {
    type uint8 {
      range "0..128";
    }
    mandatory true;
    description
      "The length of the subnet prefix.";
  }
  leaf origin {
    type ip-address-origin;
    description
      "The origin of this address.";
  }
  leaf status {
    type enumeration {
      enum preferred {
        description
          "This is a valid address that can appear as the
            destination or source address of a packet.";
      }
      enum deprecated {
        description
          "This is a valid but deprecated address that should
            no longer be used as a source address in new
            communications, but packets addressed to such an
            address are processed as expected.";
      }
      enum invalid {
        description
          "This isn't a valid address and it shouldn't appear
            as the destination or source address of a packet.";
      }
      enum inaccessible {
        description
          "The address is not accessible because the interface
            to which this address is assigned is not
            operational.";
      }
    }
  }
}
```



```
    }
    enum unknown {
      description
        "The status cannot be determined for some reason.";
    }
    enum tentative {
      description
        "The uniqueness of the address on the link is being
        verified.  Addresses in this state should not be
        used for general communication and should only be
        used to determine the uniqueness of the address.";
    }
    enum duplicate {
      description
        "The address has been determined to be non-unique on
        the link and so must not be used.";
    }
    enum optimistic {
      description
        "The address is available for use, subject to
        restrictions, while its uniqueness on a link is
        being verified.";
    }
  }
  description
    "The status of an address.  Most of the states correspond
    to states from the IPv6 Stateless Address
    Autoconfiguration protocol.";
  reference
    "RFC 4293: Management Information Base for the
      Internet Protocol (IP)
      - IpAddressStatusTC
      RFC 4862: IPv6 Stateless Address Autoconfiguration";
}
list neighbor {
  key "ip";
  description
    "A list of mappings from IPv6 addresses to
    link-layer addresses.

    This list represents the Neighbor Cache.";
  reference
    "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)";

  leaf ip {
    type inet:ipv6-address-no-zone;
    description
```

```
        "The IPv6 address of the neighbor node.";
    }
    leaf link-layer-address {
        type yang:phys-address;
        description
            "The link-layer address of the neighbor node.";
    }
    leaf origin {
        type neighbor-origin;
        description
            "The origin of this neighbor entry.";
    }
    leaf is-router {
        type empty;
        description
            "Indicates that the neighbor node acts as a router.";
    }
    leaf state {
        type enumeration {
            enum incomplete {
                description
                    "Address resolution is in progress and the link-layer
                    address of the neighbor has not yet been
                    determined.";
            }
            enum reachable {
                description
                    "Roughly speaking, the neighbor is known to have been
                    reachable recently (within tens of seconds ago).";
            }
            enum stale {
                description
                    "The neighbor is no longer known to be reachable but
                    until traffic is sent to the neighbor, no attempt
                    should be made to verify its reachability.";
            }
            enum delay {
                description
                    "The neighbor is no longer known to be reachable, and
                    traffic has recently been sent to the neighbor.
                    Rather than probe the neighbor immediately, however,
                    delay sending probes for a short while in order to
                    give upper-layer protocols a chance to provide
                    reachability confirmation.";
            }
            enum probe {
                description
                    "The neighbor is no longer known to be reachable, and
```

```

        unicast Neighbor Solicitation probes are being sent
            to verify reachability.";
    }
}
description
    "The Neighbor Unreachability Detection state of this
        entry.";
reference
    "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)
        Section 7.3.2";
}
}
}
}
}
<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-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

6. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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:

ipv4/enabled and ipv6/enabled: These leafs are used to enable or disable IPv4 and IPv6 on a specific interface. By enabling a protocol on an interface, an attacker might be able to create an unsecured path into a node (or through it if routing is also enabled). By disabling a protocol on an interface, an attacker might be able to force packets to be routed through some other interface or deny access to some or all of the network via that protocol.

ipv4/address and ipv6/address: These lists specify the configured IP addresses on an interface. By modifying this information, an attacker can cause a node to either ignore messages destined to it or accept (at least at the IP layer) messages it would otherwise ignore. The use of filtering or security associations may reduce the potential damage in the latter case.

ipv4/forwarding and ipv6/forwarding: These leafs allow a client to enable or disable the forwarding functions on the entity. By disabling the forwarding functions, an attacker would possibly be able to deny service to users. By enabling the forwarding functions, an attacker could open a conduit into an area. This might result in the area providing transit for packets it shouldn't or might allow the attacker access to the area bypassing security safeguards.

ipv6/autoconf: The leafs in this branch control the autoconfiguration of IPv6 addresses and in particular whether temporary addresses are used or not. By modifying the corresponding leafs, an attacker might impact the addresses used by a node and thus indirectly the privacy of the users using the

node.

ipv4/mtu and ipv6/mtu: Setting these leafs to very small values can be used to slow down interfaces.

7. Acknowledgments

The author wishes to thank Jeffrey Lange, Ladislav Lhotka, Juergen Schoenwaelder, and Dave Thaler for their helpful comments.

8. References

8.1. Normative References

- [I-D.ietf-netmod-interfaces-cfg]
Bjorklund, M., "A YANG Data Model for Interface Configuration", draft-ietf-netmod-interfaces-cfg-12 (work in progress), July 2012.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

8.2. Informative References

- [I-D.ietf-netmod-routing-cfg]
Lhotka, L., "A YANG Data Model for Routing Configuration", draft-ietf-netmod-routing-cfg-10 (work in progress), July 2012.
- [RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet

address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

- [RFC4293] Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, April 2006.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Appendix A. Example: NETCONF <get> reply

This section gives an example of a reply to the NETCONF <get> request for a device that implements the data model defined in this document.

```
<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>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>192.0.2.1</ip>
            <prefix-length>24</prefix-length>
          </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <mtu>1280</mtu>
          <address>
            <ip>2001:db8::10</ip>
            <prefix-length>32</prefix-length>
          </address>
          <dup-addr-detect-transmits>0</dup-addr-detect-transmits>
        </ipv6>
      </interface>
    </interfaces>
    <interfaces-state
      xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type>ethernetCsmacd</type>
        <!-- other parameters from ietf-interfaces omitted -->

        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <forwarding>false</forwarding>
          <mtu>1500</mtu>
          <address>
            <ip>192.0.2.1</ip>
            <prefix-length>24</prefix-length>
            <origin>static</origin>
          </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <forwarding>false</forwarding>
```

```
<mtu>1500</mtu>
<address>
  <ip>2001:db8::10</ip>
  <prefix-length>32</prefix-length>
  <origin>static</origin>
  <status>preferred</status>
</address>
<address>
  <ip>2001:db8::1:100</ip>
  <prefix-length>32</prefix-length>
  <origin>dhcp</origin>
  <status>preferred</status>
</address>
<neighbor>
  <ip>2001:db8::1</ip>
  <link-layer-address>00:01:02:03:04:05</link-layer-address>
  <origin>dynamic</origin>
  <is-router/>
  <state>reachable</state>
</neighbor>
<neighbor>
  <ip>2001:db8::4</ip>
  <origin>dynamic</origin>
  <state>incomplete</state>
</neighbor>
</ipv6>
</interface>
</interfaces-state>
</data>
</rpc-reply>
```

Author's Address

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

NETMOD
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2014

L. Lhotka
CZ.NIC
October 18, 2013

A YANG Data Model for Routing Management
draft-ietf-netmod-routing-cfg-11

Abstract

This document contains a specification of three YANG modules. Together they form the core routing data model which serves as a framework for configuring and managing a routing subsystem. It is expected that these modules will be augmented by additional YANG modules defining data models for individual routing protocols and other related functions. The core routing data model provides common building blocks for such extensions - routing instances, routes, routing information bases (RIB), routing protocols 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 April 21, 2014.

Copyright Notice

Copyright (c) 2013 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 and Notation	5
2.1. Glossary of New Terms	5
2.2. Tree Diagrams	6
2.3. Prefixes in Data Node Names	6
3. Objectives	8
4. The Design of the Core Routing Data Model	9
4.1. System-Controlled and User-Controlled List Entries	12
4.2. Simple versus Advanced Routers	13
5. Basic Building Blocks	15
5.1. Routing Instance	15
5.1.1. Parameters of IPv6 Routing Instance Interfaces	16
5.2. Route	17
5.3. Routing Information Base (RIB)	17
5.3.1. Multiple RIBs per Address Family	18
5.4. Routing Protocol	18
5.4.1. Routing Pseudo-Protocols	19
5.4.2. Defining New Routing Protocols	21
5.5. Route Filter	22
5.6. RPC Operations	23
6. Interactions with Other YANG Modules	24
6.1. Module "ietf-interfaces"	24
6.2. Module "ietf-ip"	24
7. Routing YANG Module	26
8. IPv4 Unicast Routing YANG Module	48
9. IPv6 Unicast Routing YANG Module	55
10. IANA Considerations	70
11. Security Considerations	72
12. Acknowledgments	73
13. References	74
13.1. Normative References	74
13.2. Informative References	74
Appendix A. The Complete Data Trees	75
A.1. Configuration Data	75
A.2. Operational State Data	77
Appendix B. Example: Adding a New Routing Protocol	79
Appendix C. Example: NETCONF <get> Reply	82
Appendix D. Change Log	88
D.1. Changes Between Versions -10 and -11	88
D.2. Changes Between Versions -09 and -10	88
D.3. Changes Between Versions -08 and -09	89

D.4.	Changes Between Versions -07 and -08	89
D.5.	Changes Between Versions -06 and -07	89
D.6.	Changes Between Versions -05 and -06	89
D.7.	Changes Between Versions -04 and -05	90
D.8.	Changes Between Versions -03 and -04	90
D.9.	Changes Between Versions -02 and -03	91
D.10.	Changes Between Versions -01 and -02	91
D.11.	Changes Between Versions -00 and -01	92
Author's Address	93

1. Introduction

This document contains a specification of the following YANG modules:

- o Module "ietf-routing" provides generic components of a routing data model.
- o Module "ietf-ipv4-unicast-routing" augments the "ietf-routing" module with additional data specific to IPv4 unicast.
- o Module "ietf-ipv6-unicast-routing" augments the "ietf-routing" module with additional data specific to IPv6 unicast, including the router configuration variables required by [RFC4861].

These modules together define the so-called core routing data model, which is proposed as a basis for the development of data models for configuration and management of more sophisticated routing systems. While these three modules can be directly used for simple IP devices with static routing, their main purpose is to provide essential building blocks for more complicated setups involving multiple routing protocols, multicast routing, additional address families, and advanced functions such as route filtering or policy routing. To this end, it is expected that the core routing data model 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 [RFC6241]:

- o client
- o message
- o protocol operation
- o server

The following terms are defined in [RFC6020]:

- o augment
- o configuration data
- o data model
- o data node
- o feature
- o mandatory node
- o module
- o state data
- o RPC operation

2.1. Glossary of New Terms

active route: a route that is actually used for sending packets. If there are multiple candidate routes with a matching destination prefix, then it is up to the routing algorithm to select the active route (or several active routes in the case of multi-path routing).

core routing data model: YANG data model resulting from the combination of "ietf-routing", "ietf-ipv4-unicast-routing" and "ietf-ipv6-unicast-routing" modules.

direct route: a route to a directly connected network.

routing information base (RIB): An object containing routes together with other information. See Section 5.3 for details.

system-controlled entry: An entry of a list in operational state data ("config false") that is created by the system independently of what has been explicitly configured. See Section 4.1 for details.

user-controlled entry: An entry of a list in operational state data ("config false") that is created and deleted as a direct consequence of certain configuration changes. See Section 4.1 for details.

2.2. Tree Diagrams

A simplified graphical representation of the complete data tree is presented in Appendix A, and similar diagrams of its various subtrees appear in the main text. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "list" or "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2.3. Prefixes in Data Node Names

In this document, names of data nodes, RPC methods and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
if	ietf-interfaces	[YANG-IF]
ip	ietf-ip	[YANG-IP]
rt	ietf-routing	Section 7
v4ur	ietf-ipv4-unicast-routing	Section 8
v6ur	ietf-ipv6-unicast-routing	Section 9
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and corresponding YANG modules

3. Objectives

The initial design of the core routing data model was driven by the following 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 information bases (RIB) 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 three YANG modules. The first module, "ietf-routing", defines the generic components of a routing system. The other two modules, "ietf-ipv4-unicast-routing" and "ietf-ipv6-unicast-routing", augment the "ietf-routing" module with additional data nodes that are needed for IPv4 and IPv6 unicast routing, respectively. Figures 1 and 2 show abridged views of the configuration and operational state data hierarchies. See Appendix A for the complete data trees.

```

+--rw routing
|   +--rw routing-instance* [name]
|   |   +--rw name
|   |   +--rw routing-instance-id?
|   |   +--rw type?
|   |   +--rw enabled?
|   |   +--rw router-id?
|   |   +--rw description?
|   |   +--rw default-ribs
|   |   |   +--rw default-rib* [address-family]
|   |   |   |   +--rw address-family
|   |   |   |   +--rw name
|   |   +--rw interfaces
|   |   |   +--rw interface* [name]
|   |   |   |   +--rw name
|   |   |   |   +--rw v6ur:ipv6-router-advertisements
|   |   |   |   ...
|   |   +--rw routing-protocols
|   |   |   +--rw routing-protocol* [name]
|   |   |   |   +--rw name
|   |   |   |   +--rw description?
|   |   |   |   +--rw enabled?
|   |   |   |   +--rw type
|   |   |   |   +--rw connected-ribs
|   |   |   |   |   ...
|   |   |   +--rw static-routes
|   |   |   |   ...
|   +--rw ribs
|   |   +--rw rib* [name]
|   |   |   +--rw name
|   |   |   +--rw id?
|   |   |   +--rw address-family
|   |   |   +--rw description?
|   |   |   +--rw recipient-ribs
|   |   |   |   +--rw recipient-rib* [rib-name]
|   |   |   |   ...
|   +--rw route-filters
|   |   +--rw route-filter* [name]
|   |   |   +--rw name
|   |   |   +--rw description?
|   |   |   +--rw type

```

Figure 1: Configuration data hierarchy.

```

+--ro routing-state
  +--ro routing-instance* [id]
    | +--ro id
    | +--ro name?
    | +--ro type?
    | +--ro router-id?
    | +--ro default-ribs
    |   | +--ro default-rib* [address-family]
    |   |   | +--ro address-family
    |   |   | +--ro rib-id
    |   +--ro interfaces
    |     | +--ro interface* [name]
    |     |   | +--ro name
    |     |   | +--ro v6ur:ipv6-router-advertisements
    |     |   | ...
    |   +--ro routing-protocols
    |     | +--ro routing-protocol* [name]
    |     |   | +--ro name
    |     |   | +--ro type
    |     |   | +--ro connected-ribs
    |     |   | ...
    +--ro ribs
    | +--ro rib* [id]
    |   | +--ro id
    |   | +--ro name?
    |   | +--ro address-family
    |   | +--ro routes
    |   |   | +--ro route* [id]
    |   |   |   | ...
    |   +--ro recipient-ribs
    |     | +--ro recipient-rib* [rib-id]
    |     | ...
    +--ro route-filters
    | +--ro route-filter* [name]
    |   | +--ro name
    |   | +--ro type

```

Figure 2: Operational state data hierarchy.

As can be seen from Figures 1 and 2, the core routing data model introduces several generic components of a routing framework: routing instances, RIBs containing lists of routes, routing protocols and route filters. The following subsections describe these components in more detail.

By combining the components in various ways, and possibly augmenting them with appropriate contents defined in other modules, various routing systems can be realized.

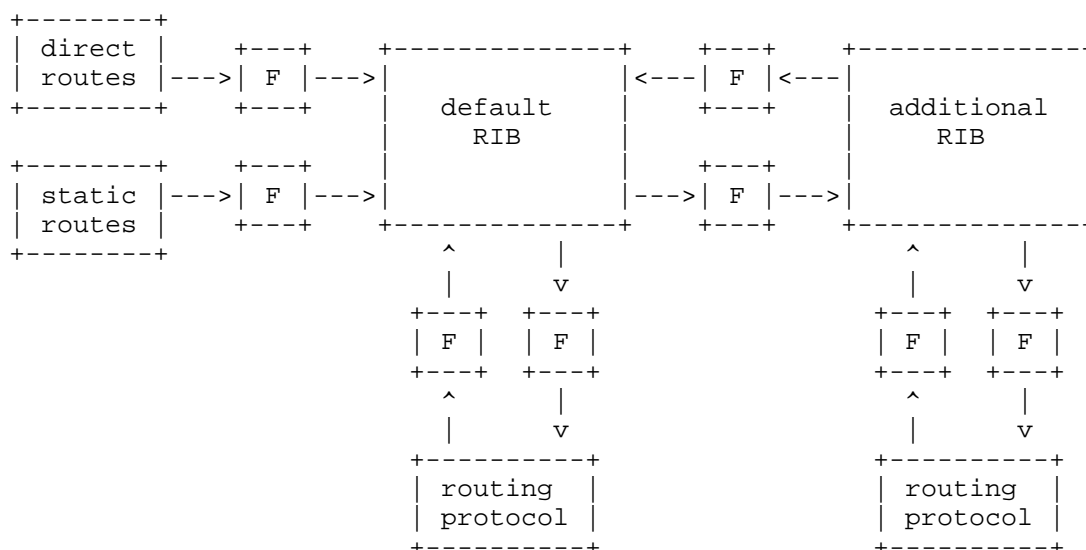


Figure 3: Example setup of a routing system

The example in Figure 3 shows a typical (though certainly not the only possible) organization of a more complex routing subsystem for a single address family. Several of its features are worth mentioning:

- o Along with the default RIB, which is always present, an additional RIB is configured.
- o Each routing protocol instance, including the "static" and "direct" pseudo-protocols, is connected to exactly one RIB with which it can exchange routes (in both directions, except for the "static" and "direct" pseudo-protocols).
- o RIBs may also be connected to each other and exchange routes in either direction (or both).
- o Route exchanges along all connections may be controlled by means of route filters, denoted by "F" in Figure 3.

4.1. System-Controlled and User-Controlled List Entries

The core routing data model defines several lists, for example "rt: routing-instance" or "rt:rib", that have to be populated with at least one entry in any properly functioning device, and additional entries may be configured by the user.

In such a list, the server creates the required item as a so-called

system-controlled entry in operational state data, i.e., inside the "rt:routing-state" container.

Additional entries may be created in the configuration by the user via the NETCONF protocol. These are the so-called user-controlled entries. If the server accepts a configured user-controlled entry, then this entry also appears in the operational state version of the list.

Each version of the list (in operational state data and configuration) uses its own set of list keys. In operational state, the keys are unique numeric identifiers assigned by the server. In configuration, the list keys are selected by the user.

The user may also provide supplemental configuration of system-controlled entries. To do so, the user creates a new entry in the configuration with an arbitrary key and desired configuration contents. In order to bind this entry with the corresponding entry in the operational state list, the user writes the operational state key as a value of a special leaf that is defined in the data model for this purpose.

An example can be seen in Appendix C: the "/routing-state/routing-instance" list has a single system-controlled entry whose "id" key has the value "1415926535". This entry is configured by the "/routing/routing-instance" entry whose "name" key is "rtr0". The binding with the operational state entry is established through the value of the leaf "routing-instance-id".

Deleting a user-controlled entry from the configuration list results in the removal of the corresponding entry in the operational state list. In contrast, if a system-controlled entry is deleted from the configuration list, only the extra configuration specified in that entry is removed but the corresponding operational state entry remains in the list.

4.2. Simple versus Advanced Routers

The core routing data model attempts to address devices with elementary routing functions as well as advanced routers. For simple devices, some parts and options of the data model are not needed and represent unnecessary complications for the implementation. Therefore, the core routing data model makes the advanced functionality optional by means of a feature "advanced-router".

Specifically, the following objects and options are supported only in devices that advertise the "advanced-router" feature:

- o multiple RIBs per address family, and user-controlled RIB entries in particular,
- o routing protocols connected to non-default RIBs,
- o RIBs configured as receivers of routes from other RIBs,
- o routes with multiple nexthops.

See the "ietf-routing" module for details.

5. Basic Building Blocks

This section describes the essential components of the core routing data model.

5.1. Routing Instance

Each routing instance in the core routing data model represents a logical router. The exact semantics of this term are left to implementations. For example, routing instances may be completely isolated virtual routers or, alternatively, they may internally share certain information.

A routing instance together with its operational status is represented as an entry of the list `"/routing-state/routing-instance"`, and identified by a unique numeric identifier. Configuration of that router instance appears as entry of the list `"/routing/routing-instance"` whose key is a routing instance name selected by the client.

An implementation MAY support multiple types of logical routers simultaneously. Instances of all routing instance types are organized as entries of the same flat `"routing-instance"` list. In order to discriminate routing instances belonging to different types, the `"type"` leaf is defined as a child of the `"routing-instance"` node.

An implementation MAY create one or more system-controlled routing instances, and MAY also pose restrictions on allowed routing instance types and on the number of supported instances for each type. For example, a simple router implementation may support only one system-controlled routing instance of the default type `"rt:standard-routing-instance"` and may not allow creation of any user-controlled instances.

Each network layer interface has to be assigned to one or more routing instances in order to be able to participate in packet forwarding, routing protocols and other operations of those routing instances. The assignment is accomplished by placing a corresponding (system- or user-controlled) entry in the list of routing instance interfaces (`"rt:interface"`). The key of the list entry is the name of a configured network layer interface, see the `"ietf-interfaces"` module [YANG-IF].

In YANG terms, the list of routing instance interfaces is modeled as the `"list"` node rather than `"leaf-list"` in order to allow for adding, via augmentation, other configuration or state data related to the corresponding interface.

Implementations MAY specify additional rules for the assignment of interfaces to routing instances. For example, it may be required that the sets of interfaces assigned to different routing instances be disjoint.

5.1.1. Parameters of IPv6 Routing Instance Interfaces

The module "ietf-ipv6-unicast-routing" augments the definition of the data node "rt:interface", in both configuration and operational state data, with definitions of the following variables as required by [RFC4861], sec. 6.2.1:

- o send-advertisements,
- o max-rtr-adv-interval,
- o min-rtr-adv-interval,
- o managed-flag,
- o other-config-flag,
- o link-mtu,
- o reachable-time,
- o retrans-timer,
- o cur-hop-limit,
- o default-lifetime,
- o prefix-list: a list of prefixes to be advertised.

The following parameters are associated with each prefix in the list:

- * valid-lifetime,
- * on-link-flag,
- * preferred-lifetime,
- * autonomous-flag.

The definitions and descriptions of the above parameters can be found in the text of the module "ietf-ipv6-unicast-routing" (Section 9).

NOTES:

1. The "IsRouter" flag, which is also required by [RFC4861], is implemented in the "ietf-ip" module [YANG-IP] (leaf "ip:forwarding").
2. The original specification [RFC4861] allows the implementations to decide whether the "valid-lifetime" and "preferred-lifetime" parameters remain the same in consecutive advertisements, or decrement in real time. However, the latter behavior seems problematic because the values might be reset again to the (higher) configured values after a configuration is reloaded. Moreover, no implementation is known to use the decrementing behavior. The "ietf-ipv6-unicast-routing" module therefore assumes the former behavior with constant values.

5.2. Route

Routes are basic elements of information in a routing system. The core routing data model 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 or action: outgoing interface, IP address of one or more adjacent routers to which a packet should be forwarded, or other special action.

The above list of route attributes suffices 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 both in configuration data, for example as manually configured static routes, and in operational state data, for example as entries in RIBs.

5.3. Routing Information Base (RIB)

A routing information base (RIB) is a list of routes complemented with administrative data, namely:

- o "source-protocol": type of the routing protocol from which the route was originally obtained.

- o "last-updated": the date and time when the route was last updated, or inserted into the RIB.

Each RIB MUST contain only routes of the same address family. In the data model, address family is represented with an identity derived from the "rt:address-family" base identity.

In the core routing data model, RIBs are operational state data represented as entries of the list "/routing-state/ribs/rib". The contents of RIBs are controlled and manipulated by routing protocol operations which may result in route additions, removals and modifications. This also includes manipulations via the "static" and/or "direct" pseudo-protocols, see Section 5.4.1.

RIBs are global, which means that a RIB may be used by any or all routing instances. However, an implementation MAY specify rules and restrictions for sharing RIBs among routing instances.

Each routing instance must have, for every supported address family, one RIB selected as the so-called default RIB. This selection is recorded in the list "default-rib". The role of default RIBs is explained in Section 5.4.

Simple router implementations that do not advertise the feature "advanced-router" will typically create one system-controlled RIB per supported address family, and declare it as a default RIB (via a system-controlled entry of the "default-rib" list).

5.3.1. Multiple RIBs per Address Family

More complex router implementations advertising the "advanced-router" feature support multiple RIBs per address family that can be used for policy routing and other purposes. Every RIB can then serve as a source of routes for other RIBs of the same address family. To achieve this, one or more recipient RIBs may be specified in the configuration of the source RIB. Optionally, a route filter may be configured for any or all recipient RIBs. Such a route filter then selects and/or manipulates the routes that are passed between the source and recipient RIB.

A RIB MUST NOT appear among its own recipient RIBs.

5.4. Routing Protocol

The core routing data model provides an open-ended framework for defining multiple routing protocol instances within a routing instance. Each routing protocol instance MUST be assigned a type, which is an identity derived from the "rt:routing-protocol" base

identity. The core routing data model defines two identities for the direct and static pseudo-protocols (Section 5.4.1).

Each routing protocol instance is connected to exactly one RIB for each address family that the routing protocol instance supports. Routes learned from the network by a routing protocol are normally installed into the connected RIB(s) and, conversely, routes from the connected RIB(s) are normally injected into the routing protocol. However, routing protocol implementations MAY specify rules that restrict this exchange of routes in either direction (or both directions).

On devices supporting the "advanced-router" feature, any RIB (system-controlled or user-controlled) may be connected to a routing protocol instance by configuring a corresponding entry in the "connected-rib" list. If such an entry is not configured for an address family, then the default RIB MUST be used as the connected RIB for this address family.

In addition, two independent route filters (see Section 5.5) may be configured for each connected RIB to apply user-defined policies controlling the exchange of routes in both directions between the routing protocol instance and the connected RIB:

- o import filter controls which routes are passed from the routing protocol instance to the connected RIB,
- o export filter controls which routes the routing protocol instance receives from the connected RIB.

Note that the terms import and export are used from the viewpoint of a RIB.

5.4.1. Routing Pseudo-Protocols

The core routing data model defines two special routing protocol types - "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 RIB (subject to route filters, if any), but an exchange in the opposite direction is not allowed.

Every routing instance MUST implement exactly one instance of the "direct" pseudo-protocol type. The name of this instance MUST also be "direct". It is the source of direct routes for all configured address families. Direct routes are normally supplied by the operating system kernel, based on the configuration of network

interface addresses, see Section 6.2. The "direct" pseudo-protocol MUST always be connected to the default RIBs of all supported address families. Unlike other routing protocol types, this connection cannot be changed in the configuration. Direct routes MAY be filtered before they appear in the default RIB.

A pseudo-protocol of the type "static" allows for specifying routes manually. It MAY be configured in zero or multiple instances, although a typical configuration will have exactly one instance per routing instance.

Static routes are configured within the "static-routes" container, see Figure 4.

```

+---rw static-routes
+---rw v4ur:ipv4
|   +---rw v4ur:route* [id]
|   |   +---rw v4ur:id
|   |   +---rw v4ur:description?
|   |   +---rw v4ur:destination-prefix
|   |   +---rw (nexthop-options)
|   |   |   +---:(special-nexthop)
|   |   |   |   +---rw v4ur:special-nexthop?
|   |   |   +---:(simple-nexthop)
|   |   |   |   +---rw v4ur:gateway?
|   |   |   |   +---rw v4ur:outgoing-interface?
|   |   +---:(nexthop-list) {rt:advanced-router}?
|   |   +---rw v4ur:nexthop* [id]
|   |   |   +---rw v4ur:id
|   |   |   +---rw v4ur:address?
|   |   |   +---rw v4ur:outgoing-interface?
|   |   |   +---rw v4ur:priority?
|   |   |   +---rw v4ur:weight?
+---rw v6ur:ipv6
+---rw v6ur:route* [id]
+---rw v6ur:id
+---rw v6ur:description?
+---rw v6ur:destination-prefix
+---rw (nexthop-options)
+---:(special-nexthop)
|   +---rw v6ur:special-nexthop?
+---:(simple-nexthop)
|   +---rw v6ur:gateway?
|   +---rw v6ur:outgoing-interface?
+---:(nexthop-list) {rt:advanced-router}?
+---rw v6ur:nexthop* [id]
+---rw v6ur:id
+---rw v6ur:address?
+---rw v6ur:outgoing-interface?
+---rw v6ur:priority?
+---rw v6ur:weight?

```

Figure 4: Structure of "static-routes" subtree.

5.4.2. Defining New Routing Protocols

It is expected that future YANG modules will create data models for additional routing protocol types. Such a new module has to define the protocol-specific configuration and state data, and it has to fit it into the core routing framework in the following way:

- o A new identity **MUST** be defined for the routing protocol and its base identity **MUST** be set to "rt:routing-protocol", or to an identity derived from "rt:routing-protocol".
- o Additional route attributes **MAY** be defined, preferably in one place by means of defining a YANG grouping. The new attributes have to be inserted as state data by augmenting the definitions of the nodes

/rt:ribs/rt:rib/rt:route

and

/rt:active-route/rt:output/rt:route,

and possibly other places in the configuration, state data and RPC input or output.

- o Configuration parameters and/or state data for the new protocol can be defined by augmenting the "routing-protocol" data node under both "/routing" and "/routing-state".
- o Per-interface configuration, including activation of the routing protocol on individual interfaces, can use references to entries in the list of routing instance interfaces (rt:interface).

By using the "when" statement, the augmented configuration parameters and state data specific to the new protocol **SHOULD** be made conditional and valid only if the value of "rt:type" or "rt:source-protocol" is equal to the new protocol's identity. It is also **RECOMMENDED** that the protocol-specific data be encapsulated in appropriately named containers.

The above steps are implemented by the example YANG module for the RIP routing protocol in Appendix B.

5.5. Route Filter

The core routing data model 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 connected RIB, or between a source and a recipient RIB. Route filters may also manipulate routes, i.e., add, delete, or modify their attributes.

Route filters are global, which means that a configured route filter may be used by any or all routing instances. However, an implementation **MAY** specify rules and restrictions for sharing route filters among routing instances.

By itself, the route filtering framework defined in this document allows for applying only two extreme routing policies which are represented by the following pre-defined route filter types:

- o "deny-all-route-filter": all routes are blocked,
- o "allow-all-route-filter": all routes are permitted.

The latter type is equivalent to no route filter.

It is expected that more comprehensive route filtering frameworks will be developed separately.

Each route filter is identified by a unique name. Its type MUST be specified by the "type" identity reference - this opens the space for multiple route filtering framework implementations.

5.6. RPC Operations

The "ietf-routing" module defines two RPC operations:

- o active-route: query the routing system for the active route(s) that are currently used for sending datagrams to a destination host whose address is passed as an input parameter.
- o route-count: retrieve the total number of entries in a RIB.

6. Interactions with Other YANG Modules

The semantics of the core routing data model also depend on several configuration parameters that are defined in other YANG modules.

6.1. Module "ietf-interfaces"

The following boolean switch is defined in the "ietf-interfaces" YANG module [YANG-IF]:

```
/if:interfaces/if:interface/if:enabled
```

If this switch is set to "false" for a network layer interface, the device **MUST** behave exactly as if that interface was not assigned to any routing instance at all.

6.2. Module "ietf-ip"

The following boolean switches are defined in the "ietf-ip" YANG module [YANG-IP]:

```
/if:interfaces/if:interface/ip:ipv4/ip:enabled
```

If this switch is set to "false" for a network layer interface, then all IPv4 routing functions related to that interface **MUST** be disabled.

```
/if:interfaces/if:interface/ip:ipv4/ip:forwarding
```

If this switch is set to "false" for a network layer interface, then the forwarding of IPv4 datagrams to and from this interface **MUST** be disabled. However, the interface may participate in other IPv4 routing functions, such as routing protocols.

```
/if:interfaces/if:interface/ip:ipv6/ip:enabled
```

If this switch is set to "false" for a network layer interface, then all IPv6 routing functions related to that interface **MUST** be disabled.

```
/if:interfaces/if:interface/ip:ipv6/ip:forwarding
```

If this switch is set to "false" for a network layer interface, then the forwarding of IPv6 datagrams to and from this interface **MUST** be disabled. However, the interface may participate in other IPv6 routing functions, such as routing protocols.

In addition, the "ietf-ip" module allows for configuring IPv4 and

IPv6 addresses and network prefixes or masks on network layer interfaces. Configuration of these parameters on an enabled interface MUST result in an immediate creation of the corresponding direct route. The destination prefix of this route is set according to the configured IP address and network prefix/mask, and the interface is set as the outgoing interface for that route.

7. Routing YANG Module

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number and all occurrences of the revision date below with the date of RFC publication (and remove this note).

```
<CODE BEGINS> file "ietf-routing@2013-10-18.yang"

module ietf-routing {

    namespace "urn:ietf:params:xml:ns:yang:ietf-routing";

    prefix "rt";

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

    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@nic.cz>
        ";

    description
        "This YANG module defines essential components for the management
        of a routing subsystem.

        Copyright (c) 2013 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.

```
";

revision 2013-10-18 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Routing Management";
}

/* Features */

feature advanced-router {
  description
    "This feature indicates that the device supports advanced
    routing functions, namely:

    - user-defined RIBs,

    - multi-path routes.

    Devices that do not support this feature MUST provide exactly
    one system-controlled RIB per supported address family. These
    RIBs then appear as entries of the list
    /routing-state/ribs/rib.
    ";
}

/* Identities */

identity address-family {
  description
    "Base identity from which identities describing address
    families are derived.";
}

identity ipv4 {
  base address-family;
  description
    "This identity represents IPv4 address family.";
}
```



```
identity ipv6 {
  base address-family;
  description
    "This identity represents IPv6 address family.";
}

identity routing-instance-type {
  description
    "Base identity from which identities describing routing
    instance types are derived.

    It is primarily intended for discriminating among different
    types of logical routers or router virtualization.
    ";
}

identity standard-routing-instance {
  base routing-instance-type;
  description
    "This identity represents a default routing instance.";
}

identity routing-protocol {
  description
    "Base identity from which routing protocol identities are
    derived.";
}

identity direct {
  base routing-protocol;
  description
    "Routing pseudo-protocol which provides routes to directly
    connected networks.";
}

identity static {
  base routing-protocol;
  description
    "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
      "Route filter that blocks all routes.";
  }

  identity allow-all-route-filter {
    base route-filter;
    description
      "Route filter that permits all routes.";
  }

  /* Type Definitions */

  typedef routing-instance-ref {
    type leafref {
      path "/rt:routing/rt:routing-instance/rt:name";
    }
    description
      "This type is used for leafs that reference a routing instance
      configuration.";
  }

  typedef routing-instance-state-ref {
    type leafref {
      path "/rt:routing-state/rt:routing-instance/rt:id";
    }
    description
      "This type is used for leafs that reference state data of a
      routing instance.";
  }

  typedef rib-ref {
    type leafref {
      path "/rt:routing/rt:ribs/rt:rib/rt:name";
    }
    description
      "This type is used for leafs that reference a RIB
      configuration.";
  }

  typedef rib-state-ref {
    type leafref {
      path "/rt:routing-state/rt:ribs/rt:rib/rt:id";
    }
    description
      "This type is used for leafs that reference a RIB in state
      data.";
  }
```

```
typedef route-filter-ref {
  type leafref {
    path "/rt:routing/rt:route-filters/rt:route-filter/rt:name";
  }
  description
    "This type is used for leafs that reference a route filter
    configuration.";
}

typedef route-filter-state-ref {
  type leafref {
    path "/rt:routing-state/rt:route-filters/rt:route-filter/"
      + "rt:name";
  }
  description
    "This type is used for leafs that reference a route filter in
    state data.";
}

/* Groupings */

grouping address-family {
  description
    "This grouping provides a leaf identifying an address
    family.";
  leaf address-family {
    type identityref {
      base address-family;
    }
    mandatory "true";
    description
      "Address family.";
  }
}

grouping router-id {
  description
    "This grouping provides the definition of router ID.";
  leaf router-id {
    type yang:dotted-quad;
    description
      "Router ID - 32-bit number in the form of a dotted quad.";
  }
}

grouping outgoing-interface {
  description
    "This grouping defines the outgoing interface for use in
```

```
        nexthops.";
    leaf outgoing-interface {
        type leafref {
            path "/routing-state/routing-instance/interfaces/interface/"
                + "name";
        }
        description
            "Name of the outgoing interface.";
    }
}

grouping special-nexthop {
    description
        "This grouping provides the leaf for specifying special nexthop
        options.";
    leaf special-nexthop {
        type enumeration {
            enum blackhole {
                description
                    "Silently discard the packet.";
            }
            enum unreachable {
                description
                    "Discard the packet and notify the sender with an error
                    message indicating that the destination host is
                    unreachable.";
            }
            enum prohibit {
                description
                    "Discard the packet and notify the sender with an error
                    message indicating that the communication is
                    administratively prohibited.";
            }
            enum receive {
                description
                    "The packet will be received by the local network
                    device.";
            }
        }
        description
            "Special nexthop options.";
    }
}

grouping nexthop-classifiers {
    description
        "This grouping provides two nexthop classifiers.";
    leaf priority {
```

```
type enumeration {
  enum primary {
    value "1";
    description
      "Primary nexthop.";
  }
  enum backup {
    value "2";
    description
      "Backup nexthop.";
  }
}
default "primary";
description
  "Simple priority for distinguishing between primary and
  backup nexthops.

  Backup nexthops are used if and only if no primary nexthops
  are reachable.
  ";
}
leaf weight {
  type uint8;
  must ". = 0 or not(.../nexthop/weight = 0)" {
    error-message "Illegal combination of zero and non-zero "
      + "nexthop weights.";
  }
  description
    "Nexthop weights must be either all zero (equal
    load-balancing) or all non-zero.";
}
default "0";
description
  "This parameter specifies the weight of the nexthop for load
  balancing. The number specifies the relative fraction of the
  traffic that will use the corresponding nexthop.

  The default value of 0 represents equal load-balancing.

  If both primary and backup nexthops are present, then the
  weights for each priority level are used separately.
  ";
}
}

grouping nexthop-content {
  description
    "Generic parameters of nexthops in routes.";
  choice nexthop-options {
```

```
    mandatory "true";
    description
      "Options for expressing the nexthop in routes.";
    case special-nexthop {
      uses special-nexthop;
    }
    case simple-nexthop {
      uses outgoing-interface;
    }
    case nexthop-list {
      if-feature advanced-router;
      list nexthop {
        key "id";
        description
          "An entry of a nexthop list.";
        leaf id {
          type uint64;
          description
            "A numeric identifier of the entry, assigned by the
             server.";
        }
        uses outgoing-interface;
        uses nexthop-classifiers;
      }
    }
  }
}

grouping route-metadata {
  description
    "Route metadata.";
  leaf source-protocol {
    type identityref {
      base routing-protocol;
    }
    mandatory "true";
    description
      "Type of the routing protocol from which the route
       originated.";
  }
  leaf last-updated {
    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 into the RIB.";
  }
}
```

```
/* Operational state data */

container routing-state {
  config "false";
  description
    "Operational state of the routing subsystem.";
  list routing-instance {
    key "id";
    description
      "Each list entry is a container for operational state data of
      a routing instance.

      An implementation MAY create one or more system-controlled
      instances, other user-controlled instances MAY be created by
      configuration.
      ";
    leaf id {
      type uint64;
      description
        "Unique numeric identifier of the routing instance.";
    }
    leaf name {
      type leafref {
        path "/routing/routing-instance/name";
      }
      description
        "The name of the routing instance assigned in the
        corresponding configuration entry (if any).
        ";
    }
    leaf type {
      type identityref {
        base routing-instance-type;
      }
      default "rt:standard-routing-instance";
      description
        "The routing instance type, primarily intended for
        discriminating among different types of logical routers,
        route virtualization, master-slave arrangements etc.,
        while keeping all routing instances in the same flat list.
        ";
    }
  }
  uses router-id {
    description
      "Global router ID.

      An implementation may choose a value if none is
      configured.
```

```
        Routing protocols MAY override this global parameter.
        ";
    }
    container default-ribs {
        description
            "Default RIBs used by the routing instance.";
        list default-rib {
            key "address-family";
            description
                "Each list entry specifies the default RIB for one
                address family.

                The default RIB is operationally connected to all
                routing protocols for which a connected RIB has not been
                explicitly configured.

                The 'direct' pseudo-protocol is always connected to the
                default RIBs.
                ";
            uses address-family;
            leaf rib-id {
                type rib-state-ref;
                mandatory "true";
                description
                    "Name of an existing RIB to be used as the default RIB
                    for the given routing instance and address family.";
            }
        }
    }
}
container interfaces {
    description
        "Network layer interfaces belonging to the routing
        instance.";
    list interface {
        key "name";
        description
            "List of network layer interfaces assigned to the routing
            instance.";
        leaf name {
            type if:interface-state-ref;
            description
                "A reference to the name of a configured network layer
                interface.";
        }
    }
}
container routing-protocols {
    description
```



```
    "Container for the list of routing protocol instances.";
list routing-protocol {
  key "name";
  description
    "Operational state of a routing protocol instance.
    ";
  leaf name {
    type string;
    description
      "The name of the routing protocol instance.";
  }
  leaf type {
    type identityref {
      base routing-protocol;
    }
    mandatory "true";
    description
      "Type of the routing protocol.";
  }
  container connected-ribs {
    if-feature advanced-router;
    description
      "Container for connected RIBs.
      ";
    list connected-rib {
      key "rib-id";
      description
        "List of RIBs to which the routing protocol instance
        is connected (at most one RIB per address family).
        ";
      leaf rib-id {
        type rib-state-ref;
        description
          "Name of an existing RIB.";
      }
      leaf import-filter {
        type route-filter-state-ref;
        description
          "Reference to a route filter that is used for
          filtering routes passed from this routing protocol
          instance to the RIB specified by the 'name'
          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 route-filter-state-ref;
    description
        "Reference to a route filter that is used for
        filtering routes passed from the RIB specified by
        the 'name' sibling node to this routing protocol
        instance.

        If this leaf is not present, the behavior is
        protocol-specific - typically it means that all
        routes are accepted.

        The 'direct' and 'static' pseudo-protocols accept
        no routes from any RIB."
};
}
}
}
}
container ribs {
    description
        "Container for RIBs.";
    list rib {
        key "id";
        description
            "Each entry represents a RIB identified by the 'name' key.
            All routes in a RIB MUST belong to the same address
            family.

            The server MUST create the default RIB for each address
            family, and MAY create other RIBs. Additional RIBs MAY be
            created in the configuration."
        };
        leaf id {
            type uint64;
            description
                "Unique numeric identifier of the RIB instance."
        }
        leaf name {
            type leafref {
                path "/routing/ribs/rib/name";
            }
            description
                "The name of the RIB assigned in the corresponding
                configuration entry (if any).";
        }
    }
}
```

```
    uses address-family;
    container routes {
        description
            "Current contents of the RIB.";
        list route {
            key "id";
            description
                "A RIB route entry. This data node MUST be augmented
                with information specific for routes of each address
                family.";
            leaf id {
                type uint64 {
                    range "1..max";
                }
                description
                    "Unique numeric identifier of the route.";
            }
            uses nexthop-content;
            uses route-metadata;
        }
    }
    container recipient-ribs {
        if-feature advanced-router;
        description
            "Container for recipient RIBs.";
        list recipient-rib {
            key "rib-id";
            description
                "List of RIBs that receive routes from this RIB.";
            leaf rib-id {
                type rib-state-ref;
                description
                    "The name of the recipient RIB.";
            }
            leaf filter {
                type route-filter-state-ref;
                description
                    "A route filter which is applied to the routes passed
                    to the recipient RIB.";
            }
        }
    }
}

container route-filters {
    description
        "Container for 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
         RIB and vice versa, or between two RIBs.

         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 type {
        type identityref {
            base route-filter;
        }
        mandatory "true";
        description
            "Type of the route-filter - an identity derived from the
             'route-filter' base identity.";
    }
}
}
}

/* Configuration Data */

container routing {
    description
        "Configuration parameters for the routing subsystem.";
    list routing-instance {
        key "name";
        unique "routing-instance-id";
        description
            "Configuration of a routing instance.
            ";
        leaf name {
            type string;
            description
                "The name of the configured routing instance.";
        }
        leaf routing-instance-id {
            type uint64;
            description
                "Reference to a system-assigned numeric identifier of the
                 routing instance."
            
```

```
        This leaf is essential for creating new configuration
        entries that refer to existing system-controlled routing
        instances.
    ";
}
leaf type {
    type identityref {
        base routing-instance-type;
    }
    default "rt:standard-routing-instance";
    description
        "The type of the routing instance.";
}
leaf enabled {
    type boolean;
    default "true";
    description
        "Enable/disable the routing instance.

        If this parameter is false, the parent routing instance is
        disabled and does not appear in operational state data,
        despite any other configuration that might be present.
    ";
}
uses router-id {
    description
        "Configuration of the global router ID.";
}
leaf description {
    type string;
    description
        "Textual description of the routing instance.";
}
container default-ribs {
    if-feature advanced-router;
    description
        "Configuration of the default RIBs used by the routing
        instance.

        The default RIB for an addressed family if by default
        connected to all routing protocol instances supporting
        that address family, and always receives direct routes.
    ";
    list default-rib {
        must "address-family=/routing/ribs/rib[name=current()/"
            + "name]/address-family" {
            error-message "Address family mismatch.";
        }
        description
    }
}
```

```
        "The entry's address family MUST match that of the
          referenced RIB.";
      }
      key "address-family";
      description
        "Each list entry configures the default RIB for one
          address family.";
      uses address-family;
      leaf name {
        type string;
        mandatory "true";
        description
          "Name of an existing RIB to be used as the default RIB
            for the given routing instance and address family.";
      }
    }
  }
  container interfaces {
    description
      "Configuration of the routing instance's interfaces.";
    list interface {
      key "name";
      description
        "List of network layer interfaces assigned to the routing
          instance.";
      leaf name {
        type if:interface-ref;
        description
          "A reference to the name of a configured network layer
            interface.";
      }
    }
  }
  container routing-protocols {
    description
      "Configuration of routing protocol instances.";
    list routing-protocol {
      key "name";
      description
        "Each entry contains configuration of a routing protocol
          instance.";
      leaf name {
        type string;
        description
          "An arbitrary name of the routing protocol instance.";
      }
      leaf description {
        type string;
      }
    }
  }
}
```

```
    description
      "Textual description of the routing protocol
       instance.";
  }
  leaf enabled {
    type boolean;
    default "true";
    description
      "Enable/disable the routing protocol instance.

      If this parameter is false, the parent routing
      protocol instance is disabled and does not appear in
      operational state data, despite any other
      configuration that might be present.
      ";
  }
  leaf type {
    type identityref {
      base routing-protocol;
    }
    mandatory "true";
    description
      "Type of the routing protocol - an identity derived
       from the 'routing-protocol' base identity.";
  }
  container connected-ribs {
    if-feature advanced-router;
    description
      "Configuration of connected RIBs.
      ";
    list connected-rib {
      must "not(/routing/ribs/rib[name=current()/"
        + "preceding-sibling::connected-rib/"
        + "name and address-family=/routing/ribs/"
        + "rib[name=current()/name]/address-family])" {
        error-message
          "Duplicate address family for connected RIBs.";
        description
          "For each address family, there MUST NOT be more
           than one connected RIB.";
      }
      key "rib-name";
      description
        "List of RIBs to which the routing protocol instance
         is connected (at most one RIB per address family).

        If no connected RIB is configured for an address
        family, the routing protocol is connected to the
```

```

        default RIB for that address family.
        ";
    leaf rib-name {
        type rib-ref;
        must "../.../type != 'rt:direct' or "
            + "../.../.../.../default-ribs/ "
            + "default-rib/name=." {
            error-message "The 'direct' protocol can be "
                + "connected only to a default RIB.";
            description
                "For the 'direct' pseudo-protocol, the connected
                RIB must always be a default RIB.";
        }
        description
            "Name of an existing RIB.";
    }
    leaf import-filter {
        type route-filter-ref;
        description
            "Configuration of import filter.";
    }
    leaf export-filter {
        type route-filter-ref;
        description
            "Configuration of export filter.";
    }
}
}
container static-routes {
    when "../type='rt:static'" {
        description
            "This container is only valid for the 'static'
            routing protocol.";
    }
    description
        "Configuration of the 'static' pseudo-protocol.

        Address family specific modules augment this node with
        their lists of routes.
        ";
}
}
}
}
container ribs {
    description
        "Configured RIBs.";
    list rib {

```



```
key "name";
unique "id";
description
    "Each entry represents a configured RIB identified by the
    'name' key.

    Entries having the same key as a system-controlled entry
    of the list /routing-state/ribs/rib are used for
    configuring parameters of that entry. Other entries define
    additional user-controlled RIBs.
    ";
leaf name {
    type string;
    description
        "The name of the RIB.";
}
leaf id {
    type uint64;
    description
        "System-assigned numeric identifier of the RIB instance.

        This leaf is essential for creating new configuration
        entries that refer to existing system-controlled RIBs.
        ";
}
uses address-family;
leaf description {
    type string;
    description
        "Textual description of the RIB.";
}
container recipient-ribs {
    if-feature advanced-router;
    description
        "Configuration of recipient RIBs.";
    list recipient-rib {
        must "name != ../../name" {
            error-message
                "Source and recipient RIBs are identical.";
            description
                "A RIB MUST NOT appear among its recipient RIBs.";
        }
        must "/routing/ribs/rib[name=current()/name]/"
            + "address-family=../../address-family" {
            error-message "Address family mismatch.";
            description
                "Address family of the recipient RIB MUST match that
                of the source RIB.";
        }
    }
}
```

```

    }
    key "rib-name";
    description
      "Each entry configures a recipient RIB.";
    leaf rib-name {
      type rib-ref;
      description
        "The name of the recipient RIB.";
    }
    leaf filter {
      type route-filter-ref;
      description
        "A route filter which is applied to the routes passed
         to the recipient RIB.";
    }
  }
}
}
}
container route-filters {
  description
    "Configuration of route filters.";
  list route-filter {
    key "name";
    description
      "Each entry configures a named route filter.";
    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 route-filter;
      }
      mandatory "true";
      description
        "Type of the route filter..";
    }
  }
}
}
```

```
/* RPC methods */

rpc active-route {
  description
    "Return the active route that a routing-instance uses for
    sending packets to a destination address.
    ";
  input {
    leaf routing-instance-id {
      type routing-instance-state-ref;
      mandatory "true";
      description
        "Identifier of the routing instance whose forwarding
        information base is being queried.

        If the routing instance with 'id' equal to
        'routing-instance-id' doesn't exist, then this operation
        SHALL fail with error-tag 'data-missing' and error-app-tag
        'routing-instance-not-found'.
        ";
    }
    container destination-address {
      description
        "Network layer destination address.

        Address family specific modules MUST augment this
        container with a leaf named 'address'.
        ";
      uses address-family;
    }
  }
  output {
    container route {
      description
        "The active route for the specified destination.

        If the routing instance has no active route for the
        destination address, no output is returned - the server
        SHALL send an <rpc-reply> containing a single element
        <ok>.

        Address family specific modules MUST augment this list
        with appropriate route contents.
        ";
      uses address-family;
      uses nexthop-content;
      uses route-metadata;
    }
  }
}
```

```
    }  
  }  
  
  rpc route-count {  
    description  
      "Return the current number of routes in a RIB.  
  
      If the RIB with 'id' equal to 'rib-id' doesn't exist, then  
      this operation SHALL fail with error-tag 'data-missing' and  
      error-app-tag 'rib-not-found'.  
      ";  
    input {  
      leaf rib-id {  
        type rib-state-ref;  
        mandatory "true";  
        description  
          "Identifier of the RIB.";  
      }  
    }  
    output {  
      leaf number-of-routes {  
        type uint64;  
        mandatory "true";  
        description  
          "Number of routes in the RIB.";  
      }  
    }  
  }  
}  
  
<CODE ENDS>
```

8. IPv4 Unicast Routing YANG Module

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number and all occurrences of the revision date below with the date of RFC publication (and remove this note).

<CODE BEGINS> file "ietf-ipv4-unicast-routing@2013-10-18.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-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: Ladislav Lhotka  
        <mailto:lhotka@nic.cz>  
        ";  
  
    description  
        "This YANG module augments the 'ietf-routing' module with basic  
        configuration and operational state data for IPv4 unicast  
        routing.  
  
        Copyright (c) 2013 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.

";

```
revision 2013-10-18 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Routing Management";
}

/* Identities */

identity ipv4-unicast {
  base rt:ipv4;
  description
    "This identity represents the IPv4 unicast address family.";
}

/* Operational state data */

augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {
  when "../rt:address-family = 'v4ur:ipv4-unicast'" {
    description
      "This augment is valid only for IPv4 unicast.";
  }
  description
    "This leaf augments an IPv4 unicast route.";
  leaf destination-prefix {
    type inet:ipv4-prefix;
    description
      "IPv4 destination prefix.";
  }
}

augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
  + "rt:nexthop-options/rt:simple-nexthop" {
  when "../rt:address-family = 'v4ur:ipv4-unicast'" {
    description
      "This augment is valid only for IPv4 unicast.";
  }
  description
```

```
    "This leaf augments the 'simple-nexthop' case of IPv4 unicast
    routes.";
  leaf gateway {
    type inet:ipv4-address;
    description
      "IPv4 address of the gateway.";
  }
}

augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
  + "rt:nexthop-options/rt:nexthop-list/rt:nexthop" {
  when "../.../rt:address-family = 'v4ur:ipv4-unicast'" {
    description
      "This augment is valid only for IPv4 unicast.";
  }
  description
    "This leaf augments the 'nexthop-list' case of IPv4 unicast
    routes.";
  leaf address {
    type inet:ipv4-address;
    description
      "IPv4 address of the nexthop.";
  }
}

/* Configuration data */

augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
  + "rt:routing-protocol/rt:static-routes" {
  description
    "This augment defines the configuration of the 'static'
    pseudo-protocol with data specific to IPv4 unicast.";
  container ipv4 {
    description
      "Configuration of a 'static' pseudo-protocol instance
      consists of a list of routes.";
    list route {
      key "id";
      ordered-by "user";
      description
        "A user-ordered list of static routes.";
      leaf id {
        type uint32 {
          range "1..max";
        }
        description
          "Unique numeric identifier of the route."
      }
    }
  }
}
```

```
        This value is unrelated to system-assigned keys of
        routes in RIBs.
    ";
}
leaf description {
    type string;
    description
        "Textual description of the route.";
}
leaf destination-prefix {
    type inet:ipv4-prefix;
    mandatory "true";
    description
        "IPv4 destination prefix.";
}
choice nexthop-options {
    mandatory "true";
    description
        "Options for expressing the nexthop in static routes.";
    case special-nexthop {
        uses rt:special-nexthop;
    }
    case simple-nexthop {
        leaf gateway {
            type inet:ipv4-address;
            description
                "IPv4 address of the gateway.";
        }
        leaf outgoing-interface {
            type leafref {
                path "../../../../../rt:interfaces/rt:interface/"
                    + "rt:name";
            }
            description
                "Name of the outgoing interface.

                Only interfaces configured for the parent routing
                instance can be given.
        ";
        }
    }
}
case nexthop-list {
    if-feature rt:advanced-router;
    list nexthop {
        key "id";
        description
            "An entry of a nexthop list.";
        leaf id {
```



```

        type uint32;
        description
            "Unique numeric identifier of the entry.

            This value is unrelated to system-assigned keys of
            nexthops in RIBs.
            ";
    }
    leaf address {
        type inet:ipv4-address;
        description
            "IPv4 address of the nexthop.";
    }
    leaf outgoing-interface {
        type leafref {
            path "../../../../../rt:interfaces/"
                + "rt:interface/rt:name";
        }
        description
            "Name of the outgoing interface.

            Only interfaces configured for the parent routing
            instance can be given.
            ";
    }
    uses rt:nexthop-classifiers;
}
}
}
}
}
}
}

/* RPC methods */

augment "/rt:active-route/rt:input/rt:destination-address" {
    when "rt:address-family='v4ur:ipv4-unicast'" {
        description
            "This augment is valid only for IPv4 unicast.";
    }
    description
        "This leaf augments the 'rt:destination-address' parameter of
        the 'rt:active-route' operation.";
    leaf address {
        type inet:ipv4-address;
        description
            "IPv4 destination address.";
    }
}

```

```
    }

    augment "/rt:active-route/rt:output/rt:route" {
      when "rt:address-family='v4ur:ipv4-unicast'" {
        description
          "This augment is valid only for IPv4 unicast.";
      }
      description
        "This leaf augments the reply to the 'rt:active-route'
        operation.";
      leaf destination-prefix {
        type inet:ipv4-prefix;
        description
          "IPv4 destination prefix.";
      }
    }
  }

  augment "/rt:active-route/rt:output/rt:route/rt:nexthop-options/"
    + "rt:simple-nexthop" {
    when "rt:address-family='v4ur:ipv4-unicast'" {
      description
        "This augment is valid only for IPv4 unicast.";
    }
    description
      "This leaf augments the 'simple-nexthop' case in the reply to
      the 'rt:active-route' operation.";
    leaf gateway {
      type inet:ipv4-address;
      description
        "IPv4 address of the gateway.";
    }
  }

  augment "/rt:active-route/rt:output/rt:route/rt:nexthop-options/"
    + "rt:nexthop-list/rt:nexthop" {
    when "../rt:address-family='v4ur:ipv4-unicast'" {
      description
        "This augment is valid only for IPv4 unicast.";
    }
    if-feature rt:advanced-router;
    description
      "This leaf augments the 'nexthop-list' case in the reply to the
      'rt:active-route' operation.";
    leaf address {
      type inet:ipv4-address;
      description
        "IPv4 address of the nexthop.";
    }
  }
}
```

```
}  
}
```

```
<CODE ENDS>
```

9. IPv6 Unicast Routing YANG Module

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number and all occurrences of the revision date below with the date of RFC publication (and remove this note).

<CODE BEGINS> file "ietf-ipv6-unicast-routing@2013-10-18.yang"

```
module ietf-ipv6-unicast-routing {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing";  
  
    prefix "v6ur";  
  
    import ietf-routing {  
        prefix "rt";  
    }  
  
    import ietf-inet-types {  
        prefix "inet";  
    }  
  
    import ietf-interfaces {  
        prefix "if";  
    }  
  
    import ietf-ip {  
        prefix "ip";  
    }  
  
    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@nic.cz>  
        ";  
  
    description
```

"This YANG module augments the 'ietf-routing' module with basic configuration and operational state data for IPv6 unicast routing.

Copyright (c) 2013 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.

";

```
revision 2013-10-18 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for Routing Management";
}

/* Identities */

identity ipv6-unicast {
  base rt:ipv6;
  description
    "This identity represents the IPv6 unicast address family.";
}

/* Operational state data */

augment "/rt:routing-state/rt:routing-instance/rt:interfaces/"
  + "rt:interface" {
  when "/if:interfaces/if:interface[if:name=current()/rt:name]/"
    + "ip:ipv6/ip:enabled='true'" {
    description
      "This augment is only valid for router interfaces with
      enabled IPv6.";
  }
  description
    "IPv6-specific parameters of router interfaces.";
  container ipv6-router-advertisements {
    description
      "Parameters of IPv6 Router Advertisements.";
  }
}
```

```
leaf send-advertisements {
  type boolean;
  default "false";
  description
    "A flag indicating whether or not the router sends periodic
    Router Advertisements and responds to Router
    Solicitations.";
  reference
    "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
    AdvSendAdvertisements.";
}
leaf max-rtr-adv-interval {
  type uint16 {
    range "4..1800";
  }
  units "seconds";
  default "600";
  description
    "The maximum time allowed between sending unsolicited
    multicast Router Advertisements from the interface.";
  reference
    "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
    MaxRtrAdvInterval.";
}
leaf min-rtr-adv-interval {
  type uint16 {
    range "3..1350";
  }
  units "seconds";
  description
    "The minimum time allowed between sending unsolicited
    multicast Router Advertisements from the interface.

    The default value to be used operationally if this leaf is
    not configured is determined as follows:

    - if max-rtr-adv-interval >= 9 seconds, the default value
      is 0.33 * max-rtr-adv-interval;

    - otherwise it is 0.75 * max-rtr-adv-interval.
    ";
  reference
    "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
    MinRtrAdvInterval.";
}
leaf managed-flag {
  type boolean;
  default "false";
```

```
    description
      "The boolean value to be placed in the 'Managed address
        configuration' flag field in the Router Advertisement.";
    reference
      "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvManagedFlag.";
  }
  leaf other-config-flag {
    type boolean;
    default "false";
    description
      "The boolean value to be placed in the 'Other
        configuration' flag field in the Router Advertisement.";
    reference
      "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvOtherConfigFlag.";
  }
  leaf link-mtu {
    type uint32;
    default "0";
    description
      "The value to be placed in MTU options sent by the router.
        A value of zero indicates that no MTU options are sent.";
    reference
      "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvLinkMTU.";
  }
  leaf reachable-time {
    type uint32 {
      range "0..3600000";
    }
    units "milliseconds";
    default "0";
    description
      "The value to be placed in the Reachable Time field in the
        Router Advertisement messages sent by the router. The
        value zero means unspecified (by this router).";
    reference
      "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvReachableTime.";
  }
  leaf retrans-timer {
    type uint32;
    units "milliseconds";
    default "0";
    description
      "The value to be placed in the Retrans Timer field in the
        Router Advertisement messages sent by the router. The
```

```
        value zero means unspecified (by this router).";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvRetransTimer.";
}
leaf cur-hop-limit {
    type uint8;
    default "64";
    description
        "The default value to be placed in the Cur Hop Limit field
        in the Router Advertisement messages sent by the router.
        The value should be set to the current diameter of the
        Internet. The value zero means unspecified (by this
        router).

        The default SHOULD be set to the value specified in IANA
        Assigned Numbers that was in effect at the time of
        implementation.
        ";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvCurHopLimit.

        IANA: IP Parameters,
        http://www.iana.org/assignments/ip-parameters
        ";
}
leaf default-lifetime {
    type uint16 {
        range "0..9000";
    }
    units "seconds";
    description
        "The value to be placed in the Router Lifetime field of
        Router Advertisements sent from the interface, in seconds.
        MUST be either zero or between max-rtr-adv-interval and
        9000 seconds. A value of zero indicates that the router is
        not to be used as a default router. These limits may be
        overridden by specific documents that describe how IPv6
        operates over different link layers.

        If this parameter is not configured, a value of 3 *
        max-rtr-adv-interval SHOULD be used.
        ";
    reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvDefaultLifeTime.";
}
```



```
container prefix-list {
  description
    "A list of prefixes that are placed in Prefix Information
    options in Router Advertisement messages sent from the
    interface.

    By default, these are all prefixes that the router
    advertises via routing protocols as being on-link for the
    interface from which the advertisement is sent.

    The link-local prefix SHOULD NOT be included in the list
    of advertised prefixes.
    ";
  reference
    "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
    AdvPrefixList.";
  list prefix {
    key "prefix-spec";
    description
      "Advertised prefix entry with parameters.";
    leaf prefix-spec {
      type inet:ipv6-prefix;
      description
        "IPv6 address prefix.";
    }
    leaf valid-lifetime {
      type uint32;
      units "seconds";
      default "2592000";
      description
        "The value to be placed in the Valid Lifetime in the
        Prefix Information option. The designated value of all
        1's (0xffffffff) represents infinity.
        ";
      reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvValidLifetime.";
    }
    leaf on-link-flag {
      type boolean;
      default "true";
      description
        "The value to be placed in the on-link flag ('L-bit')
        field in the Prefix Information option.";
      reference
        "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
        AdvOnLinkFlag.";
    }
  }
}
```

```
    leaf preferred-lifetime {
        type uint32;
        units "seconds";
        default "604800";
        description
            "The value to be placed in the Preferred Lifetime in
            the Prefix Information option, in seconds. The
            designated value of all 1's (0xffffffff) represents
            infinity.";
        reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
            AdvPreferredLifetime.";
    }
    leaf autonomous-flag {
        type boolean;
        default "true";
        description
            "The value to be placed in the Autonomous Flag field in
            the Prefix Information option.";
        reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6) -
            AdvAutonomousFlag.";
    }
}
}
}
}

augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {
    when "../../../rt:address-family = 'v6ur:ipv6-unicast'" {
        description
            "This augment is valid only for IPv6 unicast.";
    }
    description
        "This leaf augments an IPv6 unicast route.";
    leaf destination-prefix {
        type inet:ipv6-prefix;
        description
            "IPv6 destination prefix.";
    }
}

augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
    + "rt:nexthop-options/rt:simple-nexthop" {
    when "../../../rt:address-family = 'v6ur:ipv6-unicast'" {
        description
            "This augment is valid only for IPv6 unicast.";
    }
}
```

```
description
  "This leaf augments the 'simple-nexthop' case of IPv6 unicast
  routes.";
leaf gateway {
  type inet:ipv6-address;
  description
    "IPv6 address of the gateway.";
}
}

augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route/"
  + "rt:nexthop-options/rt:nexthop-list/rt:nexthop" {
  when ".../rt:address-family = 'v6ur:ipv6-unicast'" {
    description
      "This augment is valid only for IPv6 unicast.";
  }
  description
    "This leaf augments the 'nexthop-list' case of IPv6 unicast
    routes.";
  leaf address {
    type inet:ipv6-address;
    description
      "IPv6 address of the nexthop.";
  }
}

/* Configuration data */

augment
  "/rt:routing/rt:routing-instance/rt:interfaces/rt:interface" {
  when "/if:interfaces/if:interface[if:name=current()/rt:name]/"
    + "ip:ipv6/ip:enabled='true'" {
    description
      "This augment is only valid for router interfaces with
      enabled IPv6.";
  }
  description
    "Configuration of IPv6-specific parameters of router
    interfaces.";
  container ipv6-router-advertisements {
    description
      "Configuration of IPv6 Router Advertisements.

      See the corresponding parameters under /rt:routing-state for
      detailed descriptions and references.
      ";
    leaf send-advertisements {
      type boolean;
    }
  }
}
```

```
    default "false";
    description
        "A flag indicating whether or not the router sends periodic
        Router Advertisements and responds to Router
        Solicitations.";
}
leaf max-rtr-adv-interval {
    type uint16 {
        range "4..1800";
    }
    units "seconds";
    default "600";
    description
        "The maximum time allowed between sending unsolicited
        multicast Router Advertisements from the interface.";
}
leaf min-rtr-adv-interval {
    type uint16 {
        range "3..1350";
    }
    units "seconds";
    must ". <= 0.75 * ../max-rtr-adv-interval" {
        description
            "The value MUST NOT be greater than 75 % of
            'max-rtr-adv-interval'.";
    }
    description
        "The minimum time allowed between sending unsolicited
        multicast Router Advertisements from the interface.
        ";
}
leaf managed-flag {
    type boolean;
    default "false";
    description
        "The boolean value to be placed in the 'Managed address
        configuration' flag field in the Router Advertisement.";
}
leaf other-config-flag {
    type boolean;
    default "false";
    description
        "The boolean value to be placed in the 'Other
        configuration' flag field in the Router Advertisement.";
}
leaf link-mtu {
    type uint32;
    default "0";
```

```
    description
      "The value to be placed in MTU options sent by the router.
       A value of zero indicates that no MTU options are sent.";
  }
  leaf reachable-time {
    type uint32 {
      range "0..3600000";
    }
    units "milliseconds";
    default "0";
    description
      "The value to be placed in the Reachable Time field in the
       Router Advertisement messages sent by the router. The
       value zero means unspecified (by this router).";
  }
  leaf retrans-timer {
    type uint32;
    units "milliseconds";
    default "0";
    description
      "The value to be placed in the Retrans Timer field in the
       Router Advertisement messages sent by the router. The
       value zero means unspecified (by this router).";
  }
  leaf cur-hop-limit {
    type uint8;
    default "64";
    description
      "The default value to be placed in the Cur Hop Limit field
       in the Router Advertisement messages sent by the router.
       ";
  }
  leaf default-lifetime {
    type uint16 {
      range "0..9000";
    }
    units "seconds";
    description
      "The value to be placed in the Router Lifetime field of
       Router Advertisements sent from the interface, in seconds.
       ";
  }
  container prefix-list {
    description
      "Configuration of prefixes to be placed in Prefix
       Information options in Router Advertisement messages sent
       from the interface.
```

Prefixes that are advertised by default but do not have their entries in the child 'prefix' list are advertised with the default values of all parameters.

```

";
list prefix {
  key "prefix-spec";
  description
    "Advertised prefix entry.";
  leaf prefix-spec {
    type inet:ipv6-prefix;
    description
      "IPv6 address prefix.";
  }
  choice control-adv-prefixes {
    default "advertise";
    description
      "The prefix either may be explicitly removed from the
       set of advertised prefixes, or parameters with which
       it is advertised may be specified (default case).";
    leaf no-advertise {
      type empty;
      description
        "The prefix will not be advertised.

        This can be used for removing the prefix from the
        default set of advertised prefixes.
        ";
    }
  }
  case advertise {
    leaf valid-lifetime {
      type uint32;
      units "seconds";
      default "2592000";
      description
        "The value to be placed in the Valid Lifetime in
         the Prefix Information option.";
    }
    leaf on-link-flag {
      type boolean;
      default "true";
      description
        "The value to be placed in the on-link flag
         ('L-bit') field in the Prefix Information
         option.";
    }
    leaf preferred-lifetime {
      type uint32;
      units "seconds";
    }
  }
}
```

```

        must ". <= ../valid-lifetime" {
            description
                "This value MUST NOT be greater than
                valid-lifetime.";
        }
        default "604800";
        description
            "The value to be placed in the Preferred Lifetime
            in the Prefix Information option.";
    }
    leaf autonomous-flag {
        type boolean;
        default "true";
        description
            "The value to be placed in the Autonomous Flag
            field in the Prefix Information option.";
    }
}
}
}
}
}
}
}

augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
+ "rt:routing-protocol/rt:static-routes" {
    description
        "This augment defines the configuration of the 'static'
        pseudo-protocol with data specific to IPv6 unicast.";
    container ipv6 {
        description
            "Configuration of a 'static' pseudo-protocol instance
            consists of a list of routes.";
        list route {
            key "id";
            ordered-by "user";
            description
                "A user-ordered list of static routes.";
            leaf id {
                type uint32 {
                    range "1..max";
                }
                description
                    "Unique numeric identifier of the route.

                    This value is unrelated to system-assigned keys of
                    routes in RIBs.
                    ";
            }
        }
    }
}

```

```
    }
    leaf description {
        type string;
        description
            "Textual description of the route.";
    }
    leaf destination-prefix {
        type inet:ipv6-prefix;
        mandatory "true";
        description
            "IPv6 destination prefix.";
    }
    choice nexthop-options {
        mandatory "true";
        description
            "Options for expressing the nexthop in static routes.";
        case special-nexthop {
            uses rt:special-nexthop;
        }
        case simple-nexthop {
            leaf gateway {
                type inet:ipv6-address;
                description
                    "IPv6 address of the gateway.";
            }
            leaf outgoing-interface {
                type leafref {
                    path "../../../../../rt:interfaces/rt:interface/"
                        + "rt:name";
                }
                description
                    "Name of the outgoing interface.

                    Only interfaces configured for the parent routing
                    instance can be given.

                    ";
            }
        }
    }
    case nexthop-list {
        if-feature rt:advanced-router;
        list nexthop {
            key "id";
            description
                "An entry of a nexthop list.";
            leaf id {
                type uint32;
                description
                    "Unique numeric identifier of the entry.
```



```

        This value is unrelated to system-assigned keys of
        nexthops in RIBs.
        ";
    }
    leaf address {
        type inet:ipv6-address;
        description
            "IPv6 address of the nexthop.";
    }
    leaf outgoing-interface {
        type leafref {
            path "../../../../../rt:interfaces/"
                + "rt:interface/rt:name";
        }
        description
            "Name of the outgoing interface.

            Only interfaces configured for the parent routing
            instance can be given.
        ";
    }
    uses rt:nexthop-classifiers;
}
}
}
}
}
}
}

/* RPC methods */

augment "/rt:active-route/rt:input/rt:destination-address" {
    when "rt:address-family='v6ur:ipv6-unicast'" {
        description
            "This augment is valid only for IPv6 unicast.";
    }
    description
        "This leaf augments the 'rt:destination-address' parameter of
        the 'rt:active-route' operation.";
    leaf address {
        type inet:ipv6-address;
        description
            "IPv6 destination address.";
    }
}

augment "/rt:active-route/rt:output/rt:route" {
    when "rt:address-family='v6ur:ipv6-unicast'" {

```

```
        description
            "This augment is valid only for IPv6 unicast.";
    }
    description
        "This leaf augments the reply to the 'rt:active-route'
        operation.";
    leaf destination-prefix {
        type inet:ipv6-prefix;
        description
            "IPv6 destination prefix.";
    }
}

augment "/rt:active-route/rt:output/rt:route/rt:nexthop-options/"
    + "rt:simple-nexthop" {
    when "rt:address-family='v6ur:ipv6-unicast'" {
        description
            "This augment is valid only for IPv6 unicast.";
    }
    description
        "This leaf augments the 'simple-nexthop' case in the reply to
        the 'rt:active-route' operation.";
    leaf gateway {
        type inet:ipv6-address;
        description
            "IPv6 address of the gateway.";
    }
}

augment "/rt:active-route/rt:output/rt:route/rt:nexthop-options/"
    + "rt:nexthop-list/rt:nexthop" {
    when "../rt:address-family='v6ur:ipv6-unicast'" {
        description
            "This augment is valid only for IPv6 unicast.";
    }
    if-feature rt:advanced-router;
    description
        "This leaf augments the 'nexthop-list' case in the reply to the
        'rt:active-route' operation.";
    leaf address {
        type inet:ipv6-address;
        description
            "IPv6 address of the nexthop.";
    }
}
}
```

<CODE ENDS>

10. IANA Considerations

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

This document registers the following 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.

URI: urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing

Registrant Contact: The IESG.

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

This document registers the following 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  
-----
```

```
-----  
name:          ietf-ipv6-unicast-routing  
namespace:     urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing  
prefix:        v6ur  
reference:     RFC XXXX  
-----
```

11. Security Considerations

Configuration and state data conforming to the core routing data model (defined in this document) are designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

A number of data nodes defined in the YANG modules belonging to the configuration part of the core routing data model are writable/creatable/deletable (i.e., "config true" in YANG terms, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations to these data nodes, such as "edit-config", can have negative effects on the network if the protocol operations are not properly protected.

The vulnerable "config true" subtrees and data nodes are the following:

/routing/routing-instance/interfaces/interface This list assigns a network layer interface to a routing instance and may also specify interface parameters related to routing.

/routing/routing-instance/routing-protocols/routing-protocol This list specifies the routing protocols configured on a device.

/routing/route-filters/route-filter This list specifies the configured route filters which represent administrative policies for redistributing and modifying routing information.

/routing/ribs/rib This list specifies the RIBs configured for the device.

Unauthorized access to any of these lists can adversely affect the routing subsystem of both the local device and the network. This may lead to network malfunctions, delivery of packets to inappropriate destinations and other problems.

12. Acknowledgments

The author wishes to thank Nitin Bahadur, Martin Bjorklund, Joel Halpern, Wes Hardaker, Sriganesh Kini, Andrew McGregor, Jan Medved, Xiang Li, Thomas Morin, Tom Petch, Bruno Rijsman, Juergen Schoenwaelder, Phil Shafer, Dave Thaler and Yi Yang for their helpful comments and suggestions.

13. References

13.1. Normative References

- [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.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for Network Configuration Protocol (NETCONF)", RFC 6020, September 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "NETCONF Configuration Protocol", RFC 6241, June 2011.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, July 2013.
- [YANG-IF] Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-interfaces-cfg-12 (work in progress), July 2013.
- [YANG-IP] Bjorklund, M., "A YANG Data Model for IP Management", draft-ietf-netmod-ip-cfg-10 (work in progress), August 2013.

13.2. Informative References

- [RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

Appendix A. The Complete Data Trees

This appendix presents the complete configuration and operational state data trees of the core routing data model.

See Section 2.2 for an explanation of the symbols used. Data type of every leaf node is shown near the right end of the corresponding line.

A.1. Configuration Data

```

+--rw routing
  +--rw routing-instance* [name]
    +--rw name string
    +--rw routing-instance-id? uint64
    +--rw type? identityref
    +--rw enabled? boolean
    +--rw router-id? yang:dotted-quad
    +--rw description? string
    +--rw default-ribs {advanced-router}?
      +--rw default-rib* [address-family]
        +--rw address-family identityref
        +--rw name string
    +--rw interfaces
      +--rw interface* [name]
        +--rw name if:interface-ref
        +--rw v6ur:ipv6-router-advertisements
          +--rw v6ur:send-advertisements? boolean
          +--rw v6ur:max-rtr-adv-interval? uint16
          +--rw v6ur:min-rtr-adv-interval? uint16
          +--rw v6ur:managed-flag? boolean
          +--rw v6ur:other-config-flag? boolean
          +--rw v6ur:link-mtu? uint32
          +--rw v6ur:reachable-time? uint32
          +--rw v6ur:retrans-timer? uint32
          +--rw v6ur:cur-hop-limit? uint8
          +--rw v6ur:default-lifetime? uint16
          +--rw v6ur:prefix-list
            +--rw v6ur:prefix* [prefix-spec]
              +--rw v6ur:prefix-spec inet:ipv6-prefix
              +--rw (control-adv-prefixes)?
                +--:(no-advertise)
                  +--rw v6ur:no-advertise? empty
                +--:(advertise)
                  +--rw v6ur:valid-lifetime? uint32
                  +--rw v6ur:on-link-flag? boolean
                  +--rw v6ur:preferred-lifetime? uint32
                  +--rw v6ur:autonomous-flag? boolean

```



```

+---rw routing-protocols
+---rw routing-protocol* [name]
+---rw name                string
+---rw description?        string
+---rw enabled?            boolean
+---rw type                 identityref
+---rw connected-ribs {advanced-router}?
|   +---rw connected-rib* [rib-name]
|       +---rw rib-name        rib-ref
|       +---rw import-filter?   route-filter-ref
|       +---rw export-filter?   route-filter-ref
+---rw static-routes
+---rw v4ur:ipv4
|   +---rw v4ur:route* [id]
|       +---rw v4ur:id                uint32
|       +---rw v4ur:description?      string
|       +---rw v4ur:destination-prefix inet:ipv4-prefix
|       +---rw (nexthop-options)
|           +---:(special-nexthop)
|           |   +---rw v4ur:special-nexthop? enumeration
|           +---:(simple-nexthop)
|           |   +---rw v4ur:gateway?      inet:ipv4-address
|           |   +---rw v4ur:outgoing-interface? leafref
|           +---:(nexthop-list) {rt:advanced-router}?
|               +---rw v4ur:nexthop* [id]
|                   +---rw v4ur:id                uint32
|                   +---rw v4ur:address?   inet:ipv4-address
|                   +---rw v4ur:outgoing-interface? leafref
|                   +---rw v4ur:priority?      enumeration
|                   +---rw v4ur:weight?        uint8
+---rw v6ur:ipv6
+---rw v6ur:route* [id]
+---rw v6ur:id                uint32
+---rw v6ur:description?      string
+---rw v6ur:destination-prefix inet:ipv6-prefix
+---rw (nexthop-options)
+---:(special-nexthop)
|   +---rw v6ur:special-nexthop? enumeration
+---:(simple-nexthop)
|   +---rw v6ur:gateway?      inet:ipv6-address
|   +---rw v6ur:outgoing-interface? leafref
+---:(nexthop-list) {rt:advanced-router}?
+---rw v6ur:nexthop* [id]
+---rw v6ur:id                uint32
+---rw v6ur:address?   inet:ipv6-address
+---rw v6ur:outgoing-interface? leafref
+---rw v6ur:priority?      enumeration
+---rw v6ur:weight?        uint8

```

```

+--rw ribs
|   +--rw rib* [name]
|   |   +--rw name          string
|   |   +--rw id?           uint64
|   |   +--rw address-family identityref
|   |   +--rw description?   string
|   |   +--rw recipient-ribs {advanced-router}?
|   |   |   +--rw recipient-rib* [rib-name]
|   |   |   |   +--rw rib-name    rib-ref
|   |   |   |   +--rw filter?     route-filter-ref
+--rw route-filters
|   +--rw route-filter* [name]
|   |   +--rw name          string
|   |   +--rw description?   string
|   |   +--rw type           identityref

```

A.2. Operational State Data

```

+--ro routing-state
|   +--ro routing-instance* [id]
|   |   +--ro id            uint64
|   |   +--ro name?         leafref
|   |   +--ro type?         identityref
|   |   +--ro router-id?    yang:dotted-quad
|   |   +--ro default-ribs
|   |   |   +--ro default-rib* [address-family]
|   |   |   |   +--ro address-family identityref
|   |   |   |   +--ro rib-id      rib-state-ref
|   |   +--ro interfaces
|   |   |   +--ro interface* [name]
|   |   |   |   +--ro name          if:interface-state-ref
|   |   |   |   +--ro v6ur:ipv6-router-advertisements
|   |   |   |   |   +--ro v6ur:send-advertisements?    boolean
|   |   |   |   |   +--ro v6ur:max-rtr-adv-interval?   uint16
|   |   |   |   |   +--ro v6ur:min-rtr-adv-interval?   uint16
|   |   |   |   |   +--ro v6ur:managed-flag?          boolean
|   |   |   |   |   +--ro v6ur:other-config-flag?      boolean
|   |   |   |   |   +--ro v6ur:link-mtu?              uint32
|   |   |   |   |   +--ro v6ur:reachable-time?        uint32
|   |   |   |   |   +--ro v6ur:retrans-timer?         uint32
|   |   |   |   |   +--ro v6ur:cur-hop-limit?         uint8
|   |   |   |   |   +--ro v6ur:default-lifetime?      uint16
|   |   |   |   +--ro v6ur:prefix-list
|   |   |   |   |   +--ro v6ur:prefix* [prefix-spec]
|   |   |   |   |   |   +--ro v6ur:prefix-spec      inet:ipv6-prefix
|   |   |   |   |   |   +--ro v6ur:valid-lifetime?   uint32
|   |   |   |   |   |   +--ro v6ur:on-link-flag?     boolean
|   |   |   |   |   |   +--ro v6ur:preferred-lifetime? uint32

```

```

|           +---ro v6ur:autonomous-flag?      boolean
+---ro routing-protocols
|   +---ro routing-protocol* [name]
|       +---ro name                string
|       +---ro type                identityref
|       +---ro connected-ribs {advanced-router}?
|           +---ro connected-rib* [rib-id]
|               +---ro rib-id        rib-state-ref
|               +---ro import-filter? route-filter-state-ref
|               +---ro export-filter? route-filter-state-ref
+---ro ribs
|   +---ro rib* [id]
|       +---ro id                  uint64
|       +---ro name?              leafref
|       +---ro address-family      identityref
|       +---ro routes
|           +---ro route* [id]
|               +---ro id          uint64
|               +---ro (nexthop-options)
|                   +---:(special-nexthop)
|                       +---ro special-nexthop?      enumeration
|                   +---:(simple-nexthop)
|                       +---ro outgoing-interface?    leafref
|                       +---ro v4ur:gateway?          inet:ipv4-address
|                       +---ro v6ur:gateway?          inet:ipv6-address
|                   +---:(nexthop-list) {advanced-router}?
|                       +---ro nexthop* [id]
|                           +---ro id                uint64
|                           +---ro outgoing-interface? leafref
|                           +---ro priority?          enumeration
|                           +---ro weight?            uint8
|                           +---ro v4ur:address?      inet:ipv4-address
|                           +---ro v6ur:address?      inet:ipv6-address
|               +---ro source-protocol      identityref
|               +---ro last-updated?        yang:date-and-time
|               +---ro v4ur:destination-prefix? inet:ipv4-prefix
|               +---ro v6ur:destination-prefix? inet:ipv6-prefix
|       +---ro recipient-ribs {advanced-router}?
|           +---ro recipient-rib* [rib-id]
|               +---ro rib-id        rib-state-ref
|               +---ro filter?       route-filter-state-ref
+---ro route-filters
|   +---ro route-filter* [name]
|       +---ro name    string
|       +---ro type    identityref

```

Appendix B. Example: Adding a New Routing Protocol

This appendix demonstrates how the core routing data model can be extended to support a new routing protocol. The YANG module "example-rip" shown below is intended only as an illustration rather than a real definition of a data model for the RIP routing protocol. For the sake of brevity, we do not follow all the guidelines specified in [RFC6087]. See also Section 5.4.2.

```
module example-rip {  
    namespace "http://example.com/rip";  
  
    prefix "rip";  
  
    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";  
        }  
    }  
  
    grouping route-content {  
        description  
            "This grouping defines RIP-specific route attributes.";  
        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.";  
        }  
    }  
  
    augment "/rt:routing-state/rt:ribs/rt:rib/rt:routes/rt:route" {  
        when "rt:source-protocol = 'rip:rip'" {  

```

```
        description
          "This augment is only valid for a routes whose source
            protocol is RIP.";
      }
    description
      "RIP-specific route attributes.";
    uses route-content;
  }

  augment "/rt:active-route/rt:output/rt:route" {
    description
      "RIP-specific route attributes in the output of 'active-route'
        RPC.";
    uses route-content;
  }

  augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
    + "rt:routing-protocol" {
    when "rt:type = 'rip:rip'" {
      description
        "This augment is only valid for a routing protocol instance
          of type 'rip'.";
    }
    container rip {
      description
        "RIP instance configuration.";
      container interfaces {
        description
          "Per-interface RIP configuration.";
        list interface {
          key "name";
          description
            "RIP is enabled on interfaces that have an entry in this
              list, unless 'enabled' is set to 'false' for that
              entry.";
          leaf name {
            type leafref {
              path "../..../rt:interfaces/rt:interface/"
                + "rt:name";
            }
          }
          leaf enabled {
            type boolean;
            default "true";
          }
          leaf metric {
            type rip-metric;
            default "1";
          }
        }
      }
    }
  }
```

```
    }  
  }  
}  
leaf update-interval {  
  type uint8 {  
    range "10..60";  
  }  
  units "seconds";  
  default "30";  
  description  
    "Time interval between periodic updates.";  
}  
}  
}
```

Appendix C. Example: NETCONF <get> Reply

This section contains a sample reply to the NETCONF <get> message, which could be sent by a server supporting (i.e., advertising them in the NETCONF <hello> message) the following YANG modules:

- o ietf-interfaces [YANG-IF],
- o ietf-ip [YANG-IP],
- o ietf-routing (Section 7),
- o ietf-ipv4-unicast-routing (Section 8),
- o ietf-ipv6-unicast-routing (Section 9).

We assume a simple network setup as shown in Figure 5: router "A" uses static default routes with the "ISP" router as the next hop. IPv6 router advertisements are configured only on the "eth1" interface and disabled on the upstream "eth0" interface.

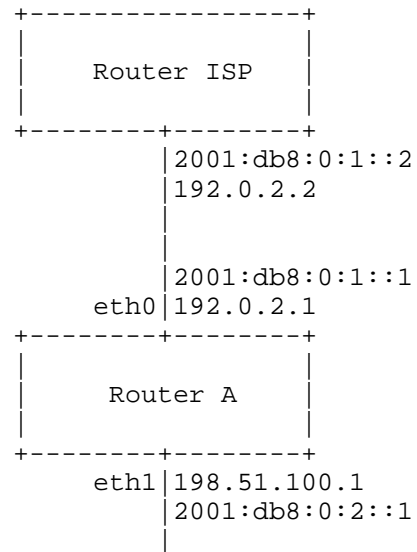


Figure 5: Example network configuration

A reply to the NETCONF <get> message sent by router "A" would then be as follows:

```
<?xml version="1.0"?>
<rpc-reply
```

```
message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:v4ur="urn:ietf:params:xml:ns:yang:ietf-ipv4-unicast-routing"
xmlns:v6ur="urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-routing"
xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
xmlns:ip="urn:ietf:params:xml:ns:yang:ietf-ip"
xmlns:rt="urn:ietf:params:xml:ns:yang:ietf-routing">
<data>
  <if:interfaces>
    <if:interface>
      <if:name>eth0</if:name>
      <if:type>ethernetCsmacd</if:type>
      <if:description>
        Uplink to ISP.
      </if:description>
      <ip:ipv4>
        <ip:address>
          <ip:ip>192.0.2.1</ip:ip>
          <ip:prefix-length>24</ip:prefix-length>
        </ip:address>
        <ip:forwarding>true</ip:forwarding>
      </ip:ipv4>
      <ip:ipv6>
        <ip:address>
          <ip:ip>2001:0db8:0:1::1</ip:ip>
          <ip:prefix-length>64</ip:prefix-length>
        </ip:address>
        <ip:forwarding>true</ip:forwarding>
        <ip:autoconf>
          <ip:create-global-addresses>>false</ip:create-global-addresses>
        </ip:autoconf>
      </ip:ipv6>
    </if:interface>
    <if:interface>
      <if:name>eth1</if:name>
      <if:type>ethernetCsmacd</if:type>
      <if:description>
        Interface to the internal network.
      </if:description>
      <ip:ipv4>
        <ip:address>
          <ip:ip>198.51.100.1</ip:ip>
          <ip:prefix-length>24</ip:prefix-length>
        </ip:address>
        <ip:forwarding>true</ip:forwarding>
      </ip:ipv4>
      <ip:ipv6>
        <ip:address>
```



```
<ip:ip>2001:0db8:0:2::1</ip:ip>
<ip:prefix-length>64</ip:prefix-length>
</ip:address>
<ip:forwarding>true</ip:forwarding>
<ip:autoconf>
  <ip:create-global-addresses>>false</ip:create-global-addresses>
</ip:autoconf>
</ip:ipv6>
</if:interface>
</if:interfaces>
<if:interfaces-state>
  <if:interface>
    <if:name>eth0</if:name>
    <if:type>ethernetCsmacd</if:type>
    <if:phys-address>00:0C:42:E5:B1:E9</if:phys-address>
    <if:oper-status>up</if:oper-status>
    <if:statistics>
      <if:discontinuity-time>
        2013-07-02T17:11:27+00:58
      </if:discontinuity-time>
    </if:statistics>
  </if:interface>
  <if:interface>
    <if:name>eth1</if:name>
    <if:type>ethernetCsmacd</if:type>
    <if:oper-status>up</if:oper-status>
    <if:phys-address>00:0C:42:E5:B1:EA</if:phys-address>
    <if:statistics>
      <if:discontinuity-time>
        2013-07-02T17:11:27+00:59
      </if:discontinuity-time>
    </if:statistics>
  </if:interface>
</if:interfaces-state>
<rt:routing>
  <rt:routing-instance>
    <rt:name>rtr0</rt:name>
    <rt:routing-instance-id>1415926535</rt:routing-instance-id>
    <rt:description>Router A</rt:description>
    <rt:interfaces>
      <rt:interface>
        <rt:name>eth1</rt:name>
        <v6ur:ipv6-router-advertisements>
          <v6ur:send-advertisements>true</v6ur:send-advertisements>
          <v6ur:prefix-list>
            <v6ur:prefix>
              <v6ur:prefix-spec>2001:db8:0:2::/64</v6ur:prefix-spec>
            </v6ur:prefix>
          </v6ur:prefix-list>
        </v6ur:ipv6-router-advertisements>
      </rt:interface>
    </rt:interfaces>
  </rt:routing-instance>
</rt:routing>
</rt:router>
```

```
    </v6ur:prefix-list>
  </v6ur:ipv6-router-advertisements>
</rt:interface>
</rt:interfaces>
<rt:routing-protocols>
  <rt:routing-protocol>
    <rt:name>st0</rt:name>
    <rt:description>
      Static routing is used for the internal network.
    </rt:description>
    <rt:type>rt:static</rt:type>
    <rt:static-routes>
      <v4ur:ipv4>
        <v4ur:route>
          <v4ur:id>1</v4ur:id>
          <v4ur:destination-prefix>0.0.0.0/0</v4ur:destination-prefix>
          <v4ur:gateway>192.0.2.2</v4ur:gateway>
        </v4ur:route>
      </v4ur:ipv4>
      <v6ur:ipv6>
        <v6ur:route>
          <v6ur:id>1</v6ur:id>
          <v6ur:destination-prefix>::/0</v6ur:destination-prefix>
          <v6ur:gateway>2001:db8:0:1::2</v6ur:gateway>
        </v6ur:route>
      </v6ur:ipv6>
    </rt:static-routes>
  </rt:routing-protocol>
</rt:routing-protocols>
</rt:routing-instance>
</rt:routing>
<rt:routing-state>
  <rt:routing-instance>
    <rt:id>1415926535</rt:id>
    <rt:name>rtr0</rt:name>
    <rt:router-id>192.0.2.1</rt:router-id>
    <rt:default-ribs>
      <rt:default-rib>
        <rt:address-family>v4ur:ipv4-unicast</rt:address-family>
        <rt:rib-id>897932384</rt:rib-id>
      </rt:default-rib>
      <rt:default-rib>
        <rt:address-family>v6ur:ipv6-unicast</rt:address-family>
        <rt:rib-id>751058209</rt:rib-id>
      </rt:default-rib>
    </rt:default-ribs>
  <rt:interfaces>
    <rt:interface>
```

```
<rt:name>eth0</rt:name>
</rt:interface>
<rt:interface>
  <rt:name>eth1</rt:name>
  <v6ur:ipv6-router-advertisements>
    <v6ur:send-advertisements>true</v6ur:send-advertisements>
    <v6ur:prefix-list>
      <v6ur:prefix>
        <v6ur:prefix-spec>2001:db8:0:2::/64</v6ur:prefix-spec>
      </v6ur:prefix>
    </v6ur:prefix-list>
  </v6ur:ipv6-router-advertisements>
</rt:interface>
</rt:interfaces>
<rt:routing-protocols>
  <rt:routing-protocol>
    <rt:name>st0</rt:name>
    <rt:type>rt:static</rt:type>
  </rt:routing-protocol>
</rt:routing-protocols>
</rt:routing-instance>
<rt:ribs>
  <rt:rib>
    <rt:id>897932384</rt:id>
    <rt:address-family>v4ur:ipv4-unicast</rt:address-family>
    <rt:routes>
      <rt:route>
        <rt:id>626433832</rt:id>
        <v4ur:destination-prefix>
          192.0.2.1/24
        </v4ur:destination-prefix>
        <rt:outgoing-interface>eth0</rt:outgoing-interface>
        <rt:source-protocol>rt:direct</rt:source-protocol>
        <rt:last-updated>2013-07-02T17:11:27+01:00</rt:last-updated>
      </rt:route>
      <rt:route>
        <rt:id>795028841</rt:id>
        <v4ur:destination-prefix>
          198.51.100.0/24
        </v4ur:destination-prefix>
        <rt:outgoing-interface>eth1</rt:outgoing-interface>
        <rt:source-protocol>rt:direct</rt:source-protocol>
        <rt:last-updated>2013-07-02T17:11:27+01:00</rt:last-updated>
      </rt:route>
      <rt:route>
        <rt:id>971693993</rt:id>
        <v4ur:destination-prefix>0.0.0.0/0</v4ur:destination-prefix>
        <rt:source-protocol>rt:static</rt:source-protocol>
```

```
<v4ur:gateway>192.0.2.2</v4ur:gateway>
  <rt:last-updated>2013-07-02T18:02:45+01:00</rt:last-updated>
</rt:route>
</rt:routes>
</rt:rib>
<rt:rib>
  <rt:id>751058209</rt:id>
  <rt:address-family>v6ur:ipv6-unicast</rt:address-family>
  <rt:routes>
    <rt:route>
      <rt:id>749445923</rt:id>
      <v6ur:destination-prefix>
        2001:db8:0:1::/64
      </v6ur:destination-prefix>
      <rt:outgoing-interface>eth0</rt:outgoing-interface>
      <rt:source-protocol>rt:direct</rt:source-protocol>
      <rt:last-updated>2013-07-02T17:11:27+01:00</rt:last-updated>
    </rt:route>
    <rt:route>
      <rt:id>78164062</rt:id>
      <v6ur:destination-prefix>
        2001:db8:0:2::/64
      </v6ur:destination-prefix>
      <rt:outgoing-interface>eth1</rt:outgoing-interface>
      <rt:source-protocol>rt:direct</rt:source-protocol>
      <rt:last-updated>2013-07-02T17:11:27+01:00</rt:last-updated>
    </rt:route>
    <rt:route>
      <rt:id>862089986</rt:id>
      <v6ur:destination-prefix>::/0</v6ur:destination-prefix>
      <v6ur:gateway>2001:db8:0:1::2</v6ur:gateway>
      <rt:source-protocol>rt:static</rt:source-protocol>
      <rt:last-updated>2013-07-02T18:02:45+01:00</rt:last-updated>
    </rt:route>
  </rt:routes>
</rt:rib>
</rt:ribs>
</rt:routing-state>
</data>
</rpc-reply>
```

Appendix D. Change Log

RFC Editor: remove this section upon publication as an RFC.

D.1. Changes Between Versions -10 and -11

- o Migrated address families from IANA enumerations to identities.
- o Terminology and node names aligned with the I2RS RIB model: router -> routing instance, routing table -> RIB.
- o Introduced uint64 keys for state lists: routing-instance, rib, route, nexthop.
- o Described the relationship between system-controlled and user-controlled list entries.
- o Feature "user-defined-routing-tables" changed into "advanced-router".
- o Made nexthop into a choice in order to allow for nexthop-list (I2RS requirement).
- o Added nexthop-list with entries having priorities (backup) and weights (load balancing).
- o Updated bibliography references.

D.2. Changes Between Versions -09 and -10

- o Added subtree for operational state data ("/routing-state").
- o Terms "system-controlled entry" and "user-controlled entry" defined and used.
- o New feature "user-defined-routing-tables". Nodes that are useful only with user-defined routing tables are now conditional.
- o Added grouping "router-id".
- o In routing tables, "source-protocol" attribute of routes now reports only protocol type, and its datatype is "identityref".
- o Renamed "main-routing-table" to "default-routing-table".

D.3. Changes Between Versions -08 and -09

- o Fixed "must" expression for "connected-routing-table".
- o Simplified "must" expression for "main-routing-table".
- o Moved per-interface configuration of a new routing protocol under 'routing-protocol'. This also affects the 'example-rip' module.

D.4. Changes Between Versions -07 and -08

- o Changed reference from RFC6021 to RFC6021bis.

D.5. Changes Between Versions -06 and -07

- o The contents of <get-reply> in Appendix C was updated: "eth[01]" is used as the value of "location", and "forwarding" is on for both interfaces and both IPv4 and IPv6.
- o The "must" expression for "main-routing-table" was modified to avoid redundant error messages reporting address family mismatch when "name" points to a non-existent routing table.
- o The default behavior for IPv6 RA prefix advertisements was clarified.
- o Changed type of "rt:router-id" to "ip:dotted-quad".
- o Type of "rt:router-id" changed to "yang:dotted-quad".
- o Fixed missing prefixes in XPath expressions.

D.6. Changes Between Versions -05 and -06

- o Document title changed: "Configuration" was replaced by "Management".
- o New typedefs "routing-table-ref" and "route-filter-ref".
- o Double slashes "//" were removed from XPath expressions and replaced with the single "/".
- o Removed uniqueness requirement for "router-id".
- o Complete data tree is now in Appendix A.
- o Changed type of "source-protocol" from "leafref" to "string".

- o Clarified the relationship between routing protocol instances and connected routing tables.
- o Added a must constraint saying that a routing table connected to the direct pseudo-protocol must not be a main routing table.

D.7. Changes Between Versions -04 and -05

- o Routing tables are now global, i.e., "routing-tables" is a child of "routing" rather than "router".
- o "must" statement for "static-routes" changed to "when".
- o Added "main-routing-tables" containing references to main routing tables for each address family.
- o Removed the defaults for "address-family" and "safi" and made them mandatory.
- o Removed the default for route-filter/type and made this leaf mandatory.
- o If there is no active route for a given destination, the "active-route" RPC returns no output.
- o Added "enabled" switch under "routing-protocol".
- o Added "router-type" identity and "type" leaf under "router".
- o Route attribute "age" changed to "last-updated", its type is "yang:date-and-time".
- o The "direct" pseudo-protocol is always connected to main routing tables.
- o Entries in the list of connected routing tables renamed from "routing-table" to "connected-routing-table".
- o Added "must" constraint saying that a routing table must not be its own recipient.

D.8. Changes Between Versions -03 and -04

- o Changed "error-tag" for both RPC methods from "missing element" to "data-missing".
- o Removed the decrementing behavior for advertised IPv6 prefix parameters "valid-lifetime" and "preferred-lifetime".

- o Changed the key of the static route lists from "seqno" to "id" because the routes needn't be sorted.
- o Added 'must' constraint saying that "preferred-lifetime" must not be greater than "valid-lifetime".

D.9. Changes Between Versions -02 and -03

- o Module "iana-afn-safi" moved to I-D "iana-if-type".
- o Removed forwarding table.
- o RPC "get-route" changed to "active-route". Its output is a list of routes (for multi-path routing).
- o New RPC "route-count".
- o For both RPCs, specification of negative responses was added.
- o Relaxed separation of router instances.
- o Assignment of interfaces to router instances needn't be disjoint.
- o Route filters are now global.
- o Added "allow-all-route-filter" for symmetry.
- o Added Section 6 about interactions with "ietf-interfaces" and "ietf-ip".
- o Added "router-id" leaf.
- o Specified the names for IPv4/IPv6 unicast main routing tables.
- o Route parameter "last-modified" changed to "age".
- o Added container "recipient-routing-tables".

D.10. Changes Between Versions -01 and -02

- o Added module "ietf-ipv6-unicast-routing".
- o The example in Appendix C now uses IP addresses from blocks reserved for documentation.
- o Direct routes appear by default in the forwarding table.

- o Network layer interfaces must be assigned to a router instance. Additional interface configuration may be present.
- o The "when" statement is only used with "augment", "must" is used elsewhere.
- o Additional "must" statements were added.
- o The "route-content" grouping for IPv4 and IPv6 unicast now includes the material from the "ietf-routing" version via "uses rt:route-content".
- o Explanation of symbols in the tree representation of data model hierarchy.

D.11. Changes Between Versions -00 and -01

- o AFN/SAFI-independent stuff was moved to the "ietf-routing" module.
- o Typedefs for AFN and SAFI were placed in a separate "iana-afn-safi" module.
- o Names of some data nodes were changed, in particular "routing-process" is now "router".
- o The restriction of a single AFN/SAFI per router was lifted.
- o RPC operation "delete-route" was removed.
- o Illegal XPath references from "get-route" to the datastore were fixed.
- o Section "Security Considerations" was written.

Author's Address

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 9, 2014

M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University
November 5, 2013

A YANG Data Model for SNMP Configuration
draft-ietf-netmod-snmp-cfg-03

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 May 9, 2014.

Copyright Notice

Copyright (c) 2013 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.	Data Model	5
2.1.	Tree Diagrams	5
2.2.	General Considerations	5
2.3.	Common Definitions	6
2.4.	Engine Configuration	6
2.5.	Target Configuration	6
2.6.	Notification Configuration	7
2.7.	Proxy Configuration	8
2.8.	Community Configuration	9
2.9.	View-based Access Control Model Configuration	10
2.10.	User-based Security Model Configuration	11
2.11.	Transport Security Model Configuration	13
2.12.	Transport Layer Security Transport Model Configuration	13
2.13.	Secure Shell Transport Model Configuration	15
3.	Implementation Guidelines	16
3.1.	Supporting read-only SNMP Access	16
3.2.	Supporting read-write SNMP access	17
4.	Definitions	18
4.1.	Module 'ietf-x509-cert-to-name'	18
4.2.	Module 'ietf-snmp'	23
4.3.	Submodule 'ietf-snmp-common'	25
4.4.	Submodule 'ietf-snmp-engine'	29
4.5.	Submodule 'ietf-snmp-target'	32
4.6.	Submodule 'ietf-snmp-notification'	36
4.7.	Submodule 'ietf-snmp-proxy'	40
4.8.	Submodule 'ietf-snmp-community'	43
4.9.	Submodule 'ietf-snmp-vacm'	47
4.10.	Submodule 'ietf-snmp-usm'	53
4.11.	Submodule 'ietf-snmp-tsm'	57
4.12.	Submodule 'ietf-snmp-tls'	60
4.13.	Submodule 'ietf-snmp-ssh'	64
5.	IANA Considerations	67
6.	Security Considerations	69
7.	Acknowledgments	71
8.	References	72
8.1.	Normative References	72
8.2.	Informative References	72
Appendix A.	Example configurations	74
A.1.	Engine Configuration Example	74
A.2.	Community Configuration Example	74
A.3.	User-based Security Model Configuration Example	75
A.4.	Target and Notification Configuration Example	76
A.5.	Proxy Configuration Example	78
A.6.	View-based Access Control Model Configuration Example	80
A.7.	Transport Layer Security Transport Model Configuration	

Example	82
Authors' Addresses	84

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], [RFC3584], [RFC5591], [RFC5592], and [RFC6353] 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.

This document also defines a YANG data model for mapping a X.509 certificate to a name.

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. Data Model

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.

2.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2.2. 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.

The persistency models in SNMP and NETCONF are quite different. In NETCONF, the persistency is defined by the datastore, whereas in SNMP it is defined either explicitly in the data model, or on a row-by-row basis by using the TEXTUAL-CONVENTION "StorageType". Thus, in the YANG model defined here, the "StorageType" columns are not present. For implementation guidelines, see Section 3.

In SNMP, row creation and deletion are controlled by using the TEXTUAL-CONVENTION "RowStatus". In NETCONF, creation and deletion are handled by the protocol, not in the data model. Thus, in the YANG model defined here, the "RowStatus" columns are not present.

2.3. Common Definitions

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

2.4. 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
    +--rw enable-authen-traps?    boolean

```

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. In this submodule, SNMP over UDP is defined. TLS and Datagram Transport Layer Security (DTLS) are also supported, defined in "ietf-snmp-tls" (Section 2.12). The "listen" 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.

2.5. 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 `"transport"` choice. In this submodule, SNMP over UDP is defined. TLS and DTLS are also supported, defined in `"ietf-snmp-tls"` (Section 2.12). The `"transport"` choice is expected to be augmented for other transports.

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 2.8), `"ietf-snmp-usm"` (Section 2.10), and `"ietf-snmp-tls"` (Section 2.12).

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 2.7).

2.6. 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       snmp:identifier
    |   +--rw type?     enumeration
  +--rw notify-filter-profile* [name]
    +--rw name          snmp:identifier
    +--rw include*      wildcard-object-identifier
    +--rw exclude*      wildcard-object-identifier

```

It also augments the "target" list defined in the "ietf-snmp-target" submodule (Section 2.5) 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.

This submodule defines the feature "notification-filter". A server implements this feature if it supports SNMP notification filtering.

2.7. 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?  snmp:identifier
    +--rw multiple-target-out? snmp:identifier

```

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

Like the `"target"` list (Section 2.5), 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 2.8), `"ietf-snmp-usm"` (Section 2.10), and `"ietf-snmp-tls"` (Section 2.12).

This submodule defines the feature `"proxy"`. A server implements this feature if it can act as an SNMP Proxy.

2.8. 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?          snmp:identifier

```

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 security-name      snmp:security-name
    | | +--:(v2c)
    | | | +--rw v2c
    | | |   +--rw security-name      snmp:security-name
    | +--rw mms?      union
  +--rw proxy
    +--rw params-in
      +--rw params
        +--:(v1)
        | +--rw v1
        |   +--rw security-name      snmp:security-name
        +--:(v2c)
        | +--rw v2c
        |   +--rw security-name      snmp:security-name

```

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

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

2.9. 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?        view-name
        +--rw write-view?       view-name
        +--rw notify-view?      view-name
    +--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" that maps to "vacmSecurityToGroupTable", and a sublist "access" that 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 a more readable representation of the "vacmViewTreeFamilyTable".

2.10. 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
|   |   |   |   +-- rw key      string
|   |   +--:(sha)
|   |   |   +--rw sha
|   |   |   |   +-- rw key      string
+--rw priv!
|   +--rw (protocol)
|   |   +--:(des)
|   |   |   +--rw des
|   |   |   |   +-- rw key      string
|   |   +--:(aes)
|   |   |   +--rw aes
|   |   |   |   +-- 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. However, the localized key can be changed. This implies that if the engine id is changed, all users keys need to be changed as well.

2.11. Transport Security Model Configuration

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

```
+--rw snmp
  +--rw tsm
    +--rw use-prefix?    boolean
```

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

```
+--rw snmp
  +--rw target* [name]
    ...
    | +--rw (params)?
    |   +--:(tsm)
    |     +--rw tsm
    |       +--rw security-name      snmp:security-name
    |       +--rw security-level     security-level
  +--rw proxy* [name]
    ...
    +--rw params-in
      +--rw (params)
      +--:(tsm)
      +--rw tsm
        +--rw security-name      snmp:security-name
        +--rw security-level     security-level
```

This submodule defines the feature "tsm". A server implements this feature if it supports the Transport Security Model (tsm) [RFC5591].

2.12. Transport Layer Security Transport Model Configuration

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


```

+---rw snmp
  ...
  +---rw target* [name]
  |   ...
  |   +---rw (transport)
  |   |   ...
  |   |   +---:(tls)
  |   |   |   +---rw tls
  |   |   |   +-- {common (d)tls transport params}
  |   |   +---:(dtls)
  |   |   |   +---rw dtls
  |   |   |   +-- {common (d)tls transport params}
  +---rw tlstm
    +---rw cert-to-name* [id]
    +---rw id                uint32
    +---rw fingerprint       x509c2n:tls-fingerprint
    +---rw map-type           identityref
    +---rw name               string

```

The "{common (d)tls transport params}" are:

```

+---rw ip?                  inet:host
+---rw port?                inet:port-number
+---rw client-fingerprint?  x509c2n:tls-fingerprint
+---rw server-fingerprint?  x509c2n:tls-fingerprint
+---rw server-identity?     snmp:admin-string

```

It also augments the "/snmp/engine/listen" container with objects for the D(TLS) transport endpoints:

```

+---rw snmp
  +---rw engine
  |   ...
  |   +---rw listen
  |   |   ...
  |   |   +---rw tls* [ip port]
  |   |   |   +---rw ip      inet:ip-address
  |   |   |   +---rw port    inet:port-number
  |   |   +---rw dtls* [ip port]
  |   |   |   +---rw ip      inet:ip-address
  |   |   |   +---rw port    inet:port-number

```

This submodule defines the feature "tlstm". A server implements this feature if it supports the Transport Layer Security (TLS) Transport Model (tlstm) [RFC6353].

2.13. Secure Shell Transport Model Configuration

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

```

+--rw snmp
  ...
  +--rw target* [name]
    ...
    +--rw (transport)
      ...
      +--:(ssh)
        +--rw ssh
          +--rw ip          inet:host
          +--rw port?       inet:port-number
          +--rw username?   string

```

It also augments the "/snmp/engine/listen" container with objects for the SSH transport endpoints:

```

+--rw snmp
  +--rw engine
    ...
    +--rw listen
      ...
      +--rw ssh* [ip port]

```

This submodule defines the feature "sshtm". A server implements this feature if it supports the Secure Shell (SSH) Transport Model (sshtm) [RFC5592].

3. Implementation Guidelines

This section describes some challenges for implementations that support both the YANG models defined in this document, and either read-write or read-only SNMP access to the same data, using the standard MIB modules.

As described in Section 2.2, the persistency models in NETCONF and SNMP are quite different. This poses a challenge for an implementation to support both NETCONF and SNMP access to the same data, in particular if the data is writable over both protocols. Specifically, the configuration data may exist in some combination of the three NETCONF configuration datastores, and this data must be mapped to rows in the SNMP tables, in some SNMP contexts, with proper values for the StorageType columns.

This problem is not new; it has been handled in many implementations that support configuration of the SNMP engine over a command line interface (CLI), which normally have a persistency model similar to NETCONF.

Since there is not one solution that works for all cases, this document does not provide a recommended solution. Instead some of the challenges involved are described below.

3.1. Supporting read-only SNMP Access

If a device implements only :writable-running, it is trivial to map the contents of "running" to data in the SNMP tables, where all instances of the StorageType columns have the value "nonVolatile".

If a device implements :candidate, but not :startup, the implementation may choose to not expose the contents of the "candidate" datastore over SNMP, and map the contents of "running" as described above. As an option, the contents of "candidate" might be accessible in a separate SNMP context.

If a device implements :startup, the handling of StorageType becomes more difficult. Since the contents of "running" and "startup" might differ, data in running cannot automatically be mapped to instances with StorageType "nonVolatile". If a particular entry exists in "running" but not in "startup", its StorageType should be "volatile". If a particular entry exists in "startup", but not "running", it should not be mapped to an SNMP instance, at least not in the default SNMP context.

3.2. Supporting read-write SNMP access

If the implementation supports read-write access to data over SNMP, and specifically creation of table rows, special attention has to be given the handling of the RowStatus and StorageType columns. The problem is to determine which table rows to store in the configuration datastores, and which configuration datastore is appropriate for each row.

The SNMP tables contain a mix of configured data and operational state, and only rows with an "active" RowStatus column should be stored in a configuration datastore.

If a device implements only :writable-running, "active" rows with a "nonVolatile" StorageType column can be stored in "running". Rows with a "volatile" StorageType column are operational state.

If a device implements :candidate, but not :writable-running, all configuration changes typically go through the "candidate", even if they are done over SNMP. An implementation might have to perform some automatic commit of the "candidate" when data is written over SNMP, since there is no explicit "commit" operation in SNMP.

If a device implements :startup, "nonVolatile" rows cannot just be written to "running", they must also be copied into "startup". "volatile" rows may be treated as operational state and not copied to any datastore, or copied into "running".

4. Definitions

4.1. Module 'ietf-x509-cert-to-name'

This YANG module imports typedefs from [RFC6991].

<CODE BEGINS> file "ietf-x509-cert-to-name.yang"

```
module ietf-x509-cert-to-name {  
  
    namespace "urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name";  
    prefix x509c2n;  
  
    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: 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  
        extracting a name from a X.509 certificate.  
  
        The algorithm used to extract a name from a X.509 certificate  
        was first defined in RFC 6353.  
  
        Copyright (c) 2013 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

"RFC6353: Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)";

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

revision 2013-11-05 {

description

"Initial revision.";

reference

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

}

typedef tls-fingerprint {

type yang:hex-string {

pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2}){0,254}';

}

description

"A fingerprint value that can be used to uniquely reference other data of potentially arbitrary length.

An tls-fingerprint value is composed of a 1-octet hashing algorithm identifier followed by the fingerprint value. The first octet value identifying the hashing algorithm is taken from the IANA TLS HashAlgorithm Registry (RFC 5246). The remaining octets are filled using the results of the hashing algorithm.";

reference "SNMP-TLS-TM-MIB.SnmpTLSPFingerprint";

}

/* Identities */

identity cert-to-name {

description

"Base identity for algorithms to derive a name from a certificate.";

}

```

identity specified {
  base cert-to-name;
  description
    "Directly specifies the name to be used for the certificate.
    The value of the leaf 'name' in 'cert-to-name' list is used.";
  reference "SNMP-TLS-TM-MIB.snmpTlstmCertSpecified";
}

identity san-rfc822-name {
  base cert-to-name;
  description
    "Maps a subjectAltName's rfc822Name to a name. The local part
    of the rfc822Name is passed unaltered but the host-part of the
    name must be passed in lowercase. This mapping results in a
    1:1 correspondence between equivalent subjectAltName
    rfc822Name values and name values except that the host-part
    of the name MUST be passed in lowercase. For example, the
    rfc822Name field FooBar@Example.COM is mapped to name
    FooBar@example.com.";
  reference "SNMP-TLS-TM-MIB.snmpTlstmCertSANRFC822Name";
}

identity san-dns-name {
  base cert-to-name;
  description
    "Maps a subjectAltName's dNSName to a name after first
    converting it to all lowercase (RFC 5280 does not specify
    converting to lowercase so this involves an extra step).
    This mapping results in a 1:1 correspondence between
    subjectAltName dNSName values and the name values.";
  reference "SNMP-TLS-TM-MIB.snmpTlstmCertSANDNSName";
}

identity san-ip-address {
  base cert-to-name;
  description
    "Maps a subjectAltName's iPAddress to a name by
    transforming the binary encoded address as follows:

    1) for IPv4, the value is converted into a
       decimal-dotted quad address (e.g., '192.0.2.1').

    2) for IPv6 addresses, the value is converted into a
       32-character all lowercase hexadecimal string
       without any colon separators.

    This mapping results in a 1:1 correspondence between
    subjectAltName iPAddress values and the name values.";

```

```

    reference "SNMP-TLS-TM-MIB.snmpTlstmCertSANIpAddress";
}

identity san-any {
    base cert-to-name;
    description
        "Maps any of the following fields using the corresponding
        mapping algorithms:

```

Type	Algorithm
rfc822Name	san-rfc822-name
dNSName	san-dns-name
iPAddress	san-ip-address

The first matching subjectAltName value found in the certificate of the above types MUST be used when deriving the name. The mapping algorithm specified in the 'Algorithm' column MUST be used to derive the name.

This mapping results in a 1:1 correspondence between subjectAltName values and name values. The three sub-mapping algorithms produced by this combined algorithm cannot produce conflicting results between themselves."

```

reference "SNMP-TLS-TM-MIB.snmpTlstmCertSANAny";
}

identity common-name {
    base cert-to-name;
    description
        "Maps a certificate's CommonName to a name after converting
        it to a UTF-8 encoding. The usage of CommonNames is
        deprecated and users are encouraged to use subjectAltName
        mapping methods instead. This mapping results in a 1:1
        correspondence between certificate CommonName values and name
        values.";
    reference "SNMP-TLS-TM-MIB.snmpTlstmCertCommonName";
}

/*
 * Groupings
 */

```

```

grouping cert-to-name {
    description
        "Defines nodes for mapping certificates to names. Modules

```


that uses this grouping should describe how the resulting name is used.";

```
list cert-to-name {
  key id;
  description
    "This list defines how certificates are mapped to names.
    The name is derived by considering each cert-to-name
    list entry in order. The cert-to-name entry's fingerprint
    determines whether the list entry is a match:

    1) If the cert-to-name list entry's fingerprint value
    matches that of the presented certificate, then consider
    the list entry as a successful match.

    2) If the cert-to-name list entry's fingerprint value
    matches that of a locally held copy of a trusted CA
    certificate, and that CA certificate was part of the CA
    certificate chain to the presented certificate, then
    consider the list entry as a successful match.
```

Once a matching cert-to-name list entry has been found, the map-type is used to determine how the name associated with the certificate should be determined. See the map-type leaf's description for details on determining the name value. If it is impossible to determine a name from the cert-to-name list entry's data combined with the data presented in the certificate, then additional cert-to-name list entries MUST be searched looking for another potential match.

Security administrators are encouraged to make use of certificates with subjectAltName fields that can be mapped to names so that a single root CA certificate can allow all child certificate's subjectAltName to map directly to a name via a 1:1 transformation.";

reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNEntry";

```
leaf id {
  type uint32;
  description
    "The id specifies the order in which the entries in the
    cert-to-name list are searched. Entries with lower
    numbers are searched first.";
  reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNID";
}
```

```
leaf fingerprint {
  type x509c2n:tls-fingerprint;
```

```

        mandatory true;
        description
            "Specifies a value with which the fingerprint of the
            certificate presented by the peer is compared. If the
            fingerprint of the certificate presented by the peer does
            not match the fingerprint configured, then the entry is
            skipped and the search for a match continues.";
        reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNFFingerprint";
    }

    leaf map-type {
        type identityref {
            base cert-to-name;
        }
        mandatory true;
        description
            "Specifies the algorithm used to map the certificate
            presented by the peer to a name.

            Mappings that need additional configuration objects should
            use the 'when' statement to make them conditional based on
            the 'map-type'.";
        reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNMapType";
    }

    leaf name {
        when "../map-type = 'x509c2n:specified'";
        type string;
        mandatory true;
        description
            "Directly specifies the NETCONF username when the
            'map-type' is 'specified'.";
        reference "SNMP-TLS-TM-MIB.snmpTlstmCertToTSNData";
    }
}
}
}

```

<CODE ENDS>

4.2. Module 'ietf-snmp'

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

```

module ietf-snmp {

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

```

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

```
include ietf-snmp-common {
    revision-date 2013-11-05;
}
include ietf-snmp-engine {
    revision-date 2013-11-05;
}
include ietf-snmp-target {
    revision-date 2013-11-05;
}
include ietf-snmp-notification {
    revision-date 2013-11-05;
}
include ietf-snmp-proxy {
    revision-date 2013-11-05;
}
include ietf-snmp-community {
    revision-date 2013-11-05;
}
include ietf-snmp-usm {
    revision-date 2013-11-05;
}
include ietf-snmp-tsm {
    revision-date 2013-11-05;
}
include ietf-snmp-vacm {
    revision-date 2013-11-05;
}
include ietf-snmp-tls {
    revision-date 2013-11-05;
}
include ietf-snmp-ssh {
    revision-date 2013-11-05;
}
```

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) 2013 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 2013-11-05 {  
  description  
    "Initial revision."  
  reference  
    "RFC XXXX: A YANG Data Model for SNMP Configuration";  
}
```

}

<CODE ENDS>

4.3. Submodule 'ietf-snmp-common'

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

```
submodule ietf-snmp-common {
```

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

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:     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) 2013 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 2013-11-05 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

/* 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.";
  reference
    "RFC3411: An Architecture for Describing SNMP Management
    Frameworks";
}

typedef security-name {
  type admin-string {
    length "1..32";
  }
  description

```

```

    "The security-name type represents an SNMP security name.";
    reference
        "RFC3411: An Architecture for Describing SNMP Management
            Frameworks";
}

typedef security-model {
    type union {
        type enumeration {
            enum v1 { value 1; }
            enum v2c { value 2; }
            enum usm { value 3; }
            enum tsm { value 4; }
        }
        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 yang:hex-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., '80:00:02:b8:04:61:62:63'.";
    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>

```

4.4. Submodule 'ietf-snmp-engine'

```

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

submodule ietf-snmp-engine {

    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 engines.

Copyright (c) 2013 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 2013-11-05 {
  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 v1 {
    type empty;
  }
  leaf v2c {
    type empty;
  }
  leaf v3 {
    type empty;
  }
}

```

```

    }
  }

  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";
  }

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

```

<CODE ENDS>

4.5. 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) 2013 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 2013-11-05 {

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 does not cover 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 {
  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>

4.6. 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) 2013 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 2013-11-05 {
 description
 "Initial revision.";
 reference
 "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

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

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 snmp:identifier;
  mandatory true;
  description
    "Target tag, selects a set of notification targets.

    Implementations MAY restrict the values of this leaf
    to be one of the available values of /snmp/target/tag in
    a valid configuration.";
  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 snmp: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 snmp: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";
    }
}

}

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>
```

4.7. 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) 2013 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 2013-11-05 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

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

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 snmp:identifier;
    description
        "Implementations MAY restrict the values of this leaf
        to be one of the available values of /snmp/target/name in
        a valid configuration.";
    reference "SNMP-PROXY-MIB.snmpProxySingleTargetOut";
}
leaf multiple-target-out {
    when "../type = 'trap' or ../type = 'inform'";
    type snmp:identifier;
    description
        "Implementations MAY restrict the values of this leaf
        to be one of the available values of /snmp/target/tag in
        a valid configuration.";
    reference "SNMP-PROXY-MIB.snmpProxyMultipleTargetOut";
}

```

```
    }  
  }  
}
```

<CODE ENDS>

4.8. 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) 2013 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 2013-11-05 {
 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 snmp:identifier;
    description
        "Used to limit access for this community to the specified
        targets.

        Implementations MAY restrict the values of this leaf
        to be one of the available values of /snmp/target/tag in
        a valid configuration.";
    reference "SNMP-COMMUNITY-MIB.snmpCommunityTransportTag";
}

```



```

    }
  }

  grouping vl-target-params {
    container vl {
      description
        "SNMPv1 parameters type.
        Represents snmpTargetParamsMPModel '0',
        snmpTargetParamsSecurityModel '1', and
        snmpTargetParamsSecurityLevel 'noAuthNoPriv'.";
      leaf security-name {
        type snmp:security-name;
        mandatory true;
        description
          "Implementations MAY restrict the values of this leaf
          to be one of the available values of
          /snmp/community/security-name in a valid configuration.";
        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 security-name {
        type snmp:security-name;
        mandatory true;
        description
          "Implementations MAY restrict the values of this leaf
          to be one of the available values of
          /snmp/community/security-name in a valid configuration.";
        reference "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
      }
    }
  }

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

<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) 2013 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 2013-11-05 {

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 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.

    Implementations MAY restrict the values of this
    leaf to be one of the available values of
    /snmp/vacm/view/name in a valid configuration.";
  reference
    "SNMP-VIEW-BASED-ACM-MIB.vacmAccessReadViewName";
}

leaf write-view {
  type 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.

    Implementations MAY restrict the values of this
    leaf to be one of the available values of
    /snmp/vacm/view/name in a valid configuration.";
  reference
    "SNMP-VIEW-BASED-ACM-MIB.vacmAccessWriteViewName";
}

```

```

    leaf notify-view {
        type 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.

            Implementations MAY restrict the values of this
            leaf to be one of the available values of
            /snmp/vacm/view/name in a valid configuration.";
        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
            "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyName";
    }

    leaf-list include {
        type snmp:wildcard-object-identifier;
        description
            "A family of subtrees included in this MIB view.";
        reference
            "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilySubtree
            SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyMask
            SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyType";
    }

    leaf-list exclude {
        type snmp:wildcard-object-identifier;
        description
            "A family of subtrees excluded from this MIB view.";
        reference
            "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilySubtree

```

```
SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyMask
SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyType";
    }
  }
}

<CODE ENDS>
```

4.10. Submodule 'ietf-snmp-usm'

This YANG submodule imports YANG extensions from [RFC6536].

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

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

  import ietf-yang-types {
    prefix yang;
  }
  import ietf-netconf-acm {
    prefix nacm;
  }

  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) 2013 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 2013-11-05 {

description

"Initial revision.";

reference

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

}

grouping key {

leaf key {

type yang:hex-string;

mandatory true;

nacm:default-deny-all;

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;
      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 snmpTargetParamsMPPModel '3' and
      snmpTargetParamsSecurityModel '3'";
    leaf user-name {
      type snmp:security-name;
      mandatory true;
      reference
        "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
    }
    leaf security-level {
      type snmp: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;
      }
    }
  }
}

<CODE ENDS>
```

4.11. Submodule 'ietf-snmp-tsm'

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

submodule ietf-snmp-tsm {

  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 Transport Security Model (TSM) of SNMP.

Copyright (c) 2013 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

"RFC5591: Transport Security Model for the
Simple Network Management Protocol (SNMP)";

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

revision 2013-11-05 {

description

"Initial revision.";

reference

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

}

feature tsm {

description

"A server implements this feature if it supports the
Transport Security Model for SNMP.";

reference

"RFC5591: Transport Security Model for the
Simple Network Management Protocol (SNMP)";

}

augment /snmp:snmp {

if-feature tsm;

```

    container tsm {
      description
        "Configuration of the Transport-based Security Model";

      leaf use-prefix {
        type boolean;
        default false;
        reference
          "SNMP-TSM-MIB.snmpTsmConfigurationUsePrefix";
      }
    }
  }

  grouping tsm-target-params {
    container tsm {
      description
        "Transport based security SNMPv3 parameters type.

        Represents snmpTargetParamsMModel '3' and
        snmpTargetParamsSecurityModel '4'";

      leaf security-name {
        type snmp:security-name;
        mandatory true;
        reference
          "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
      }
      leaf security-level {
        type snmp:security-level;
        mandatory true;
        reference
          "SNMP-TARGET-MIB.snmpTargetParamsSecurityLevel";
      }
    }
  }

  augment /snmp:snmp/snmp:target/snmp:params {
    if-feature tsm;
    case tsm {
      uses tsm-target-params;
    }
  }

  augment /snmp:snmp/snmp:proxy/snmp:params-in/snmp:params {
    if-feature tsm;
    case tsm {
      uses tsm-target-params;
    }
  }
}

```

```
}
```

```
<CODE ENDS>
```

4.12. Submodule 'ietf-snmp-tls'

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

```
submodule ietf-snmp-tls {  
    belongs-to ietf-snmp {  
        prefix snmp;  
    }  
  
    import ietf-inet-types {  
        prefix inet;  
    }  
    import ietf-x509-cert-to-name {  
        prefix x509c2n;  
    }  
  
    include ietf-snmp-common;  
    include ietf-snmp-engine;  
    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 the Transport Layer Security Transport Model (TLSTM)  
        of SNMP."
```

Copyright (c) 2013 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

"RFC6353: Transport Layer Security (TLS) Transport Model for
the Simple Network Management Protocol (SNMP)";

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

revision 2013-11-05 {
 description
 "Initial revision.";
 reference
 "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

feature tlstm {
 description
 "A server implements this feature if it supports the
 Transport Layer Security Transport Model for SNMP.";
 reference
 "RFC6353: Transport Layer Security (TLS) Transport Model for
 the Simple Network Management Protocol (SNMP)";
}

augment /snmp:snmp/snmp:engine/snmp:listen {
 if-feature tlstm;
 list tls {
 key "ip port";
 description
 "A list of IPv4 and IPv6 addresses and ports to which the
 engine listens for SNMP messages over TLS.";

 leaf ip {


```

    type inet:ip-address;
    description
      "The IPv4 or IPv6 address on which the engine listens
       for SNMP messages over TLS.";
  }
  leaf port {
    type inet:port-number;
    description
      "The TCP port on which the engine listens for SNMP
       messages over TLS.";
  }
}
list dtls {
  key "ip port";
  description
    "A list of IPv4 and IPv6 addresses and ports to which the
     engine listens for SNMP messages over DTLS.";

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

augment /snmp:snmp {
  if-feature tlstm;
  container tlstm {
    uses x509c2n:cert-to-name {
      description
        "Defines how certificates are mapped to names. The
         resulting name is used as a security name.";
      refine cert-to-name/map-type {
        description
          "Mappings that use the snmpTlstmCertToTSNData column
           need to augment the 'cert-to-name' list
           with additional configuration objects corresponding
           to the snmpTlstmCertToTSNData value. Such objects
           should use the 'when' statement to make them
           conditional based on the 'map-type'.";
      }
    }
  }
}

```

```

    }
  }
}

grouping tls-transport {
  leaf ip {
    type inet:host;
    mandatory true;
    reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
              SNMP-TLS-TM-MIB.SnmpTLSAddress";
  }
  leaf port {
    type inet:port-number;
    default 10161;
    reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
              SNMP-TLS-TM-MIB.SnmpTLSAddress";
  }
  leaf client-fingerprint {
    type x509c2n:tls-fingerprint;
    reference "SNMP-TLS-TM-MIB.snmpTlstmParamsClientFingerprint";
  }
  leaf server-fingerprint {
    type x509c2n:tls-fingerprint;
    reference "SNMP-TLS-TM-MIB.snmpTlstmAddrServerFingerprint";
  }
  leaf server-identity {
    type snmp:admin-string;
    reference "SNMP-TLS-TM-MIB.snmpTlstmAddrServerIdentity";
  }
}

augment /snmp:snmp/snmp:target/snmp:transport {
  if-feature tlstm;
  case tls {
    reference "SNMP-TLS-TM-MIB.snmpTLSTCPDomain";
    container tls {
      uses tls-transport;
    }
  }
}

augment /snmp:snmp/snmp:target/snmp:transport {
  if-feature tlstm;
  case dtls {
    reference "SNMP-TLS-TM-MIB.snmpDTLSUDPDDomain";
    container dtls {
      uses tls-transport;
    }
  }
}

```

```
    }  
  }  
}
```

<CODE ENDS>

4.13. Submodule 'ietf-snmp-ssh'

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

```
submodule ietf-snmp-ssh {  
  
  belongs-to ietf-snmp {  
    prefix snmp;  
  }  
  
  import ietf-inet-types {  
    prefix inet;  
  }  
  
  include ietf-snmp-common;  
  include ietf-snmp-engine;  
  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 the Secure Shell Transport Model (SSHTM)  
    of SNMP."
```

Copyright (c) 2013 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

"RFC5592: Secure Shell Transport Model for the
Simple Network Management Protocol (SNMP)";

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

revision 2013-11-05 {
 description
 "Initial revision.";
 reference
 "RFC XXXX: A YANG Data Model for SNMP Configuration";
}

feature sshtm {
 description
 "A server implements this feature if it supports the
 Secure Shell Transport Model for SNMP.";
 reference
 "RFC5592: Secure Shell Transport Model for the
 Simple Network Management Protocol (SNMP)";
}

augment /snmp:snmp/snmp:engine/snmp:listen {
 if-feature sshtm;
 list ssh {
 key "ip port";
 description
 "A list of IPv4 and IPv6 addresses and ports to which the
 engine listens for SNMP messages over SSH.";

 leaf ip {

```

        type inet:ip-address;
        description
            "The IPv4 or IPv6 address on which the engine listens
            for SNMP messages over SSH.";
    }
    leaf port {
        type inet:port-number;
        description
            "The TCP port on which the engine listens for SNMP
            messages over SSH.";
    }
}

augment /snmp:snmp/snmp:target/snmp:transport {
    if-feature sshtm;
    case ssh {
        reference "SNMP-SSH-TM-MIB.snmpSSHDomain";
        container ssh {
            leaf ip {
                type inet:host;
                mandatory true;
                reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
                    SNMP-SSH-TM-MIB.SnmpSSHAddress";
            }
            leaf port {
                type inet:port-number;
                default 5161;
                reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
                    SNMP-SSH-TM-MIB.SnmpSSHAddress";
            }
            leaf username {
                type string;
                reference "SNMP-TARGET-MIB.snmpTargetAddrTAddress
                    SNMP-SSH-TM-MIB.SnmpSSHAddress";
            }
        }
    }
}
}
}

```

<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-snmp

Registrant Contact: The NETMOD WG of the IETF.

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

This document registers the following YANG modules in the YANG Module Names registry [RFC6020].

name:	ietf-snmp
namespace:	urn:ietf:params:xml:ns:yang:ietf-snmp
prefix:	snmp
reference:	RFC XXXX
name:	ietf-x509-cert-to-name
namespace:	urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name
prefix:	x509c2n
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-vacm
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-usm
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-tsm
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-tls
parent:	ietf-snmp
reference:	RFC XXXX
name:	ietf-snmp-ssh
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 [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

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:

- o The /snmp/engine subtree contains the configuration of general parameters of an SNMP engine such as the endpoints to listen on, the transports and SNMP versions enabled, or the engine's identity. Write access to this subtree should only be granted to entities configuring general SNMP engine parameters.
- o The /snmp/target subtree contains the configuration of SNMP targets and in particular which transports to use and their security parameters. Write access to this subtree should only be granted to the security administrator and entities configuring SNMP notification forwarding behavior.
- o The /snmp/notify and /snmp/notify-filter-profile subtrees contain the configuration for SNMP notification forwarding and filtering mechanism. Write access to this subtree should only be granted to entities configuring SNMP notification forwarding behavior.
- o The /snmp/proxy subtree contains the configuration for SNMP proxies. Write access to this subtree should only be granted to entities configuring SNMP proxies.
- o The /snmp/community subtree contains the configuration of the community-based security model. Write access to this subtree should only be granted to the security administrator.
- o The /snmp/usm subtree contains the configuration of the user-based security model. Write access to this subtree should only be granted to the security administrator.
- o The /snmp/tsm subtree contains the configuration of the transport layer security model for SNMP. Write access to this subtree should only be granted to the security administrator.

- o The /snmp/tlstm subtree contains the configuration of the SNMP transport over (D)TLS and in particular the configuration how certificates are mapped to SNMP security names. Write access to this subtree should only be granted to the security administrator.
- o The /snmp/vacm subtree contains the configuration of the view-based access control mechanism used by SNMP to authorize access to management information via SNMP. Write access to this subtree should only be granted to the security administrator.

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:

- o The /snmp/engine subtree subtree exposes general information about an SNMP engine such as which version(s) of SNMP are enabled or which transports are enabled.
- o The /snmp/target subtree exposes information which transports are used to reach certain SNMP targets which transport specific parameters are used.
- o The /snmp/notify and /snmp/notify-filter-profile subtrees exposes information how notifications are filtered and forwarded to notification targets.
- o The /snmp/proxy subtree exposes information about proxy relationships.
- o The /snmp/community, /snmp/usm, /snmp/tsm, /snmp/tlstm, and /snmp/vacm subtrees are specifically sensitive since they expose information about the authentication and authorization policy used by an SNMP engine.

7. Acknowledgments

The authors want to thank Wes Hardaker and David Spakes for their reviews and valuable comments.

Juergen Schoenwaelder was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

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.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.
- [RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

8.2. Informative References

- [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.
- [RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model for the Simple Network Management Protocol (SNMP)", RFC 5591, June 2009.
- [RFC5592] Harrington, D., Salowey, J., and W. Hardaker, "Secure Shell Transport Model for the Simple Network Management Protocol (SNMP)", RFC 5592, June 2009.
- [RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", RFC 6353, July 2011.

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>
  <target>
    <name>bluebox</name>
    <udp>
      <ip>2001:db8::abcd</ip>
      <port>161</port>
    </udp>
    <tag>blue</tag>
    <v2c>
      <security-name>community-public</security-name>
    </v2c>
  </target>
</snmp>

```

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>
    </remote>
  </usm>
</snmp>

```

```

        </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>
          <!--
            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</key>
        </aes>
      </priv>
    </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>192.0.2.3</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>192.0.2.6</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>192.0.2.9</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

Below is an XML instance document showing the configuration of a proxy forwarder application. It proxies SNMPv2c messages from command generators to a file server running a SNMPv1 agent that recognizes two community strings, "private" and "public", with different associated read views. The fileserver is represented as two "target" instances, one for each community string.

If the proxy receives a SNMPv2c message with the community string "public" from a device in the "Office Network" or "Home Office Network", it gets tagged as "trusted", and the proxy uses the "private" community string when sending the message to the file server. Other SNMPv2c messages with the community string "public" get tagged as "non-trusted", and the proxy uses the "public" community string for these messages. There is also a special "backdoor" community string that can be used from any location to get "trusted" access.

The "Office Network" and "Home Office Network" are represented as two "target" instances.

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
  <target>
    <name>File Server (private)</name>
    <udp>
      <ip>192.0.2.1</ip>
    </udp>
    <v1>
      <security-name>private</security-name>
    </v1>
  </target>
  <target>
    <name>File Server (public)</name>
    <udp>
      <ip>192.0.2.1</ip>
    </udp>
    <v1>
      <security-name>public</security-name>
    </v1>
  </target>
  <target>
    <name>Office Network</name>
    <udp>
      <ip>192.0.2.0</ip>
      <prefix-length>24</prefix-length>
    </udp>
    <tag>office</tag>
  </target>
</snmp>
```

```

</target>
<target>
  <name>Home Office Network</name>
  <udp>
    <ip>203.0.113.0</ip>
    <prefix-length>24</prefix-length>
  </udp>
  <tag>home-office</tag>
</target>

<!--
  Communities c1,c2,c3, and c4 are used for incoming messages
  that should be forwarded.

  Communities c3 and c5 are used for outgoing messages to the
  file server.
-->
<community>
  <index>c1</index>
  <security-name>public</security-name>
  <engine-id>80:00:61:81:c8</engine-id>
  <context>trusted</context>
  <target-tag>office</target-tag>
</community>
<community>
  <index>c2</index>
  <security-name>public</security-name>
  <engine-id>80:00:61:81:c8</engine-id>
  <context>trusted</context>
  <target-tag>home-office</target-tag>
</community>
<community>
  <index>c3</index>
  <security-name>public</security-name>
  <engine-id>80:00:61:81:c8</engine-id>
  <context>not-trusted</context>
</community>
<community>
  <index>c4</index>
  <text-name>backdoor</text-name>
  <security-name>public</security-name>
  <engine-id>80:00:61:81:c8</engine-id>
  <context>trusted</context>
</community>
<community>
  <index>c5</index>
  <security-name>private</security-name>
  <engine-id>80:00:61:81:c8</engine-id>

```

```

    <context>trusted</context>
  </community>

  <proxy>
    <name>p1</name>
    <type>read</type>
    <context-engine-id>80:00:61:81:c8</context-engine-id>
    <context-name>trusted</context-name>
    <params-in>
      <v2c>
        <security-name>public</security-name>
      </v2c>
    </params-in>
    <single-target-out>File Server (private)</single-target-out>
  </proxy>
  <proxy>
    <name>p2</name>
    <type>read</type>
    <context-engine-id>80:00:61:81:c8</context-engine-id>
    <context-name>not-trusted</context-name>
    <params-in>
      <v2c>
        <security-name>public</security-name>
      </v2c>
    </params-in>
    <single-target-out>File Server (public)</single-target-out>
  </proxy>
</snmp>

```

If an SNMPv2c Get request with community string "public" is received from an IP address tagged as "office" or "home-office", or if the request is received from anywhere else with community string "backdoor", the implied context is "trusted" and so proxy entry "p1" matches. The request is forwarded to the file server as SNMPv1 with community "private" using community table entry "c5" for outbound params lookup.

If an SNMPv2c Get request with community string "public" is received from any other IP address, the implied context is "not-trusted" so proxy entry "p2" matches, and the request is forwarded to the file server as SNMPv1 with community "public".

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>1.3.6.1.2.1.1</include>
      <include>1.3.6.1.2.1.11</include>
      <include>1.3.6.1.6.3.10.2.1</include>
      <include>1.3.6.1.6.3.11.2.1</include>
      <include>1.3.6.1.6.3.15.1.1</include>
    </view>
  </vacm>
</snmp>

```

A.7. Transport Layer Security Transport Model Configuration Example

Below is an XML instance document showing the configuration of the certificate to security name mapping (see Appendix A.2 and A.3 of [RFC6353]).

```
<snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp"
      xmlns:x509c2n=
        "urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name">
  <tlstm>
    <cert-to-name>
      <id>1</id>
      <fingerprint>11:0A:05:11:00</fingerprint>
      <map-type>x509c2n:san-any</map-type>
    </cert-to-name>
    <cert-to-name>
      <id>2</id>
      <fingerprint>11:0A:05:11:00</fingerprint>
      <map-type>x509c2n:specified</map-type>
      <name>
        Joe Cool
      </name>
    </cert-to-name>
  </tlstm>
</snmp>
```

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2014

A. Bierman
YumaWorks
M. Bjorklund
Tail-f Systems
July 4, 2013

YANG Data Model for System Management
draft-ietf-netmod-system-mgmt-08

Abstract

This document defines a YANG data model for the configuration and identification of some common system properties within a device containing a NETCONF server. This includes data node definitions for system identification, time-of-day management, user management, DNS resolver configuration, and some protocol operations for system management.

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 5, 2014.

Copyright Notice

Copyright (c) 2013 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
1.2. Tree Diagrams	3
2. Objectives	5
2.1. System Identification	5
2.2. System Time Management	5
2.3. User Authentication	5
2.4. DNS Resolver	5
2.5. System Control	6
3. System Data Model	7
3.1. System Identification	7
3.2. System Time Management	7
3.3. DNS Resolver Model	8
3.4. RADIUS Client Model	8
3.5. User Authentication Model	9
3.5.1. SSH Public Key Authentication	9
3.5.2. Local User Password Authentication	10
3.5.3. RADIUS Password Authentication	10
3.6. System Control	10
4. System YANG module	11
5. IANA Considerations	29
6. Security Considerations	30
7. Change Log	32
7.1. 00-01	32
7.2. 01-02	32
7.3. 02-03	32
7.4. 03-04	32
7.5. 04-05	32
7.6. 05-06	33
7.7. 06-07	33
7.8. 07-08	34
8. References	35
8.1. Normative References	35
8.2. Informative References	36
Authors' Addresses	37

1. Introduction

This document defines a YANG [RFC6020] data model for the configuration and identification of some common properties within a device containing a NETCONF server.

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

The "ietf-system" YANG module defined in this document provides the following features:

- o system identification configuration and monitoring
- o system time-of-day configuration and monitoring
- o user authentication configuration
- o local users configuration
- o DNS resolver configuration
- o system control operations (shutdown, restart, setting time)

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].

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- o Ellipsis ("...") stands for contents of subtrees that are not shown.

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 state data and the information returned by the server is 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 need to be supported. Use of one or more NTP servers to automatically set the system date and time need to be possible. Utilization of the Timezone database [RFC6557] also need to be supported. It should be possible for the server, as well as clients, to configure the system to use NTP.

2.3. User Authentication

The authentication mechanism need to support password authentication over RADIUS, to support deployment scenarios with centralized authentication servers. Additionally, local users need to 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 [RFC6242] the authentication model need to 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. It should be possible for the server, as well as clients, to configure the system authentication properties.

2.4. DNS Resolver

The configuration of the DNS resolver within the system containing the NETCONF server is required to control how domain names are resolved.

2.5. System Control

A few operations are needed to support common tasks such as restarting the device or setting the system date and time.

3. System Data Model

3.1. System Identification

The data model for system identification has the following structure:

```

+--rw system
|   +--rw contact?          string
|   +--rw hostname?        inet:domain-name
|   +--rw location?         string
+--ro system-state
|   +--ro platform
|   |   +--ro os-name?      string
|   |   +--ro os-release?   string
|   |   +--ro os-version?   string
|   |   +--ro machine?      string

```

3.2. System Time Management

The data model for system time management has the following structure:

```

+--rw system
|   +--rw clock
|   |   +--rw (timezone)?
|   |   |   +--:(timezone-location)
|   |   |   |   +--rw timezone-location?    ianatz:iana-timezone
|   |   |   +--:(timezone-utc-offset)
|   |   |   |   +--rw timezone-utc-offset?   int16
|   |   +--rw ntp
|   |   |   +--rw enabled?    boolean
|   |   |   +--rw server* [name]
|   |   |   |   +--rw name          string
|   |   |   |   +--rw (transport)
|   |   |   |   |   +--:(udp)
|   |   |   |   |   |   +--rw udp
|   |   |   |   |   |   |   +--rw address    inet:host
|   |   |   |   |   |   |   +--rw port?     inet:port-number
|   |   |   |   +--rw association-type?   enumeration
|   |   |   +--rw iburst?                boolean
|   |   |   +--rw prefer?                boolean
|   +--ro system-state
|   |   +--ro clock
|   |   |   +--ro current-datetime?       yang:date-and-time
|   |   |   +--ro boot-datetime?         yang:date-and-time

```

New "case" statements can be added over time or augmented to the "transport" choice to support other transport protocols.

3.3. DNS Resolver Model

The data model for configuration of the DNS resolver has the following structure:

```

+--rw system
  +--rw dns-resolver
    +--rw search*      inet:domain-name
    +--rw server* [name]
      | +--rw name      string
      | +--rw (transport)
      | | +--:(udp-and-tcp)
      | | +--udp-and-tcp
      | |   +--rw address      inet:ip-address
      | |   +--rw port?       inet:port-number
    +--rw options
      +--rw timeout?      uint8
      +--rw attempts?    uint8

```

New "case" statements can be added over time or augmented to the "transport" choice to support other transport protocols.

3.4. RADIUS Client Model

The data model for configuration of the RADIUS client has the following structure:

```

+--rw system
  +--rw radius
    +--rw server* [name]
      | +--rw name          string
      | +--rw (transport)
      | | +--:(udp)
      | | +--rw udp
      | |   +--rw address      inet:host
      | |   +--rw authentication-port?  inet:port-number
      | |   +--rw shared-secret  string
      | +--rw authentication-type?  identityref
    +--rw options
      +--rw timeout?      uint8
      +--rw attempts?    uint8

```

New "case" statements can be added over time or augmented to the "transport" choice to support other transport protocols.

3.5. 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 can be defined by other standard documents or by vendors.

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

The data model for user authentication has the following structure:

```
+--rw system
  +--rw authentication
    +--rw user-authentication-order*  identityref
    +--rw user* [name]
      +--rw name          string
      +--rw password?    crypt-hash
      +--rw ssh-key* [name]
        +--rw name          string
        +--rw algorithm    string
        +--rw key-data     binary
```

3.5.1. 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. 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, and verifies the key as described in [RFC4253].

3.5.2. 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.

3.5.3. RADIUS Password Authentication

If the NETCONF server advertises the "radius-authentication" feature, the device supports user authentication using 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.

3.6. System Control

Two protocol operations are included to restart or shutdown the system. The 'system-restart' operation can be used to restart the entire system (not just the NETCONF server). The 'system-shutdown' operation can be used to power off the entire system.

4. System YANG module

This YANG module imports YANG extensions from [RFC6536], and imports YANG types from [I-D.ietf-netmod-rfc6021-bis] and [I-D.ietf-netmod-iana-timezones]. It also references [RFC1035], [RFC1321], [RFC2865], [RFC3418], [RFC5607], [RFC5966], [IEEE-1003.1-2008], and [FIPS.180-3.2008].

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

<CODE BEGINS> file "ietf-system@2013-07-04.yang"

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

  import ietf-yang-types {
    prefix yang;
  }

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-netconf-acm {
    prefix nacm;
  }

  import iana-timezones {
    prefix ianatz;
  }

  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@yumaworks.com>
```

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

description

"This module contains a collection of YANG definitions for the configuration and identification of some common system properties within a device containing a NETCONF server. This includes data node definitions for system identification, time-of-day management, user management, DNS resolver configuration, and some protocol operations for system management.

Copyright (c) 2013 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.: remove this note
// Note: extracted from draft-ietf-netmod-system-mgmt-07.txt

// RFC Ed.: update the date below with the date of RFC publication
// and remove this note.
revision "2013-07-04" {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for System Management";
}

/*
 * Typedefs
 */

typedef crypt-hash {
  type string {
    pattern
      '$0$.*'
```

```

+ '|$1${a-zA-Z0-9./}{1,8}${a-zA-Z0-9./}{22}'
+ '|$5$(rounds=\d+)?[a-zA-Z0-9./]{1,16}${a-zA-Z0-9./}{43}'
+ '|$6$(rounds=\d+)?[a-zA-Z0-9./]{1,16}${a-zA-Z0-9./}{86}';
}
description

```

"The crypt-hash type is used to store passwords using a hash function. The algorithms for applying the hash function and encoding the result are implemented in various UNIX systems as the function crypt(3).

A value of this type matches one of the forms:

```

$0$<clear text password>
$<id>$<salt>$<password hash>
$<id>$<parameter>$<salt>$<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>\$' or '\$<id>\$<parameter>\$<salt>\$' is prepended to the result. This value is stored in the configuration data store.

If a value starting with '\$<id>\$', where <id> is not '0', 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 accepted.

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

```

"IEEE Std 1003.1-2008 - crypt() function
Wikipedia: http://en.wikipedia.org/wiki/Crypt_(C)
RFC 1321: The MD5 Message-Digest Algorithm
FIPS.180-3.2008: Secure Hash Standard";

```

```
}

/*
 * Features
 */

feature radius {
  description
    "Indicates that the device can be configured as a RADIUS
    client.";
  reference
    "RFC 2865: Remote Authentication Dial In User Service "
    + "(RADIUS)";
}

feature authentication {
  description
    "Indicates that the device supports configuration
    for user authentication.";
}

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

feature radius-authentication {
  if-feature radius;
  if-feature authentication;
  description
    "Indicates that the device supports configuration of user
    authentication over 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 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 ntp-udp-port {
  description
    "Indicates that the device supports the configuration of
    the UDP port for NTP servers.

    This is a 'feature' since many implementations do not support
    any other port than the default port.";
}

feature timezone-location {
  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
    RFC 6557 (BCP 175)";
}

feature dns-udp-tcp-port {
  description
    "Indicates that the device supports the configuration of
    the UDP and TCP port for DNS servers.

    This is a 'feature' since many implementations do not support
    any other port than the default port.";
```

```
}

/*
 * 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.";
}

identity radius-authentication-type {
  description
    "Base identity for RADIUS authentication types.";
}

identity radius-pap {
  base radius-authentication-type;
  description
    "The device requests PAP authentication from the RADIUS
     server.";
  reference
    "RFC 2865: Remote Authentication Dial In User Service";
}

identity radius-chap {
  base radius-authentication-type;
  description
    "The device requests CHAP authentication from the RADIUS
     server.";
```



```
    reference
      "RFC 2865: Remote Authentication Dial In User Service";
  }

/*
 * Top-level container
 */

container system {
  description
    "System group configuration.";

  leaf contact {
    type string;
    description
      "The administrator contact information for the system.
       The server MAY restrict the size and characters in
       order to maintain compatibility with the sysContact
       MIB object.";
    reference
      "RFC 3418 - Management Information Base (MIB) for the
       Simple Network Management Protocol (SNMP)
       SNMPv2-MIB.sysContact";
  }
  leaf hostname {
    type inet:domain-name;
    description
      "The name of the host. This name can be a single domain
       label, or the fully qualified domain name of the host.";
  }
  leaf location {
    type string;
    description
      "The system location. The server MAY restrict the size
       and characters in order to maintain compatibility with
       the sysLocation MIB object.";
    reference
      "RFC 3418 - Management Information Base (MIB) for the
       Simple Network Management Protocol (SNMP)
       SNMPv2-MIB.sysLocation";
  }
  container clock {
    description
      "Configuration of the system date and time properties.";

    choice timezone {
      description
        "The system timezone information.";
    }
  }
}
```

```
    case timezone-location {
      if-feature timezone-location;
      leaf timezone-location {
        type ianatz:iana-timezone;
        description
          "The TZ database location identifier string
           to use for the system, such as 'Europe/Stockholm'.";
      }
    }
  }
  case timezone-utc-offset {
    leaf timezone-utc-offset {
      type int16 {
        range "-1500 .. 1500";
      }
      units "minutes";
      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'.
         Note that automatic daylight savings time adjustment
         is not provided, if this object is used.";
    }
  }
}

container ntp {
  if-feature ntp;
  description
    "Configuration of the NTP client.";

  leaf enabled {
    type boolean;
    default true;
    description
      "Indicates that the system should attempt
       to synchronize the system clock with an
       NTP server from the 'ntp/server' list.";
  }
  list server {
    key name;
    description
      "List of NTP servers to use for
       system clock synchronization.  If '/system/ntp/enabled'
       is 'true', then the system will attempt to
       contact and utilize the specified NTP servers.";

    leaf name {
```

```
    type string;
    description
        "An arbitrary name for the NTP server.";
}
choice transport {
    mandatory true;
    description
        "The transport protocol specific parameters for this
        server.";

    case udp {
        container udp {
            description
                "Contains UDP specific configuration parameters
                for NTP.";
            leaf address {
                type inet:host;
                mandatory true;
                description
                    "The address of the NTP server.";
            }
            leaf port {
                if-feature ntp-udp-port;
                type inet:port-number;
                default 123;
                description
                    "The port number of the NTP server.";
            }
        }
    }
}
leaf association-type {
    type enumeration {
        enum server {
            description
                "Use client association mode. This device
                will not provide synchronization to the
                configured NTP server.";
        }
        enum peer {
            description
                "Use symmetric active association mode.
                This device may provide synchronization
                to the configured NTP server.";
        }
        enum pool {
            description
                "Use client association mode with one or
```

```
        more of the NTP servers found by DNS
        resolution of the domain name given by
        the 'address' leaf. This device will not
        provide synchronization to the servers.";
    }
}
default server;
description
    "The desired association type for this NTP server.";
}
leaf iburst {
    type boolean;
    default false;
    description
        "Indicates whether this server should enable burst
        synchronization or not.";
}
leaf prefer {
    type boolean;
    default false;
    description
        "Indicates whether this server should be preferred
        or not.";
}
}
}

container dns-resolver {
    description
        "Configuration of the DNS resolver.";

    leaf-list search {
        type inet:domain-name;
        ordered-by user;
        description
            "An ordered list of domains to search when resolving
            a host name.";
    }
    list server {
        key name;
        ordered-by user;
        description
            "List of the DNS servers that the resolver should query.

            When the resolver is invoked by a calling application, it
            sends the query to the first name server in this list. If
            no response has been received within 'timeout' seconds,
            the resolver continues with the next server in the list.
```

If no response is received from any server, the resolver continues with the first server again. When the resolver has traversed the list 'attempts' times without receiving any response, it gives up and returns an error to the calling application.

Implementations MAY limit the number of entries in this list.";

```
leaf name {
  type string;
  description
    "An arbitrary name for the DNS server.";
}
choice transport {
  mandatory true;
  description
    "The transport protocol specific parameters for this
    server.";

  case udp-and-tcp {
    container udp-and-tcp {
      description
        "Contains UDP and TCP specific configuration
        parameters for DNS.";
      reference
        "RFC 1035: Domain Implementation and Specification
        RFC 5966: DNS over TCP";

      leaf address {
        type inet:ip-address;
        mandatory true;
        description
          "The address of the DNS server.";
      }
      leaf port {
        if-feature dns-udp-tcp-port;
        type inet:port-number;
        default 53;
        description
          "The UDP and TCP port number of the DNS server.";
      }
    }
  }
}
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 timeout {
    type uint8 {
      range "1..max";
    }
    units "seconds";
    default "5";
    description
      "The amount of time the resolver will wait for a
        response from each remote name server before
        retrying the query via a different name server.";
  }
  leaf attempts {
    type uint8 {
      range "1..max";
    }
    default "2";
    description
      "The number of times the resolver will send a query to
        all its name servers before giving up and returning an
        error to the calling application.";
  }
}

container radius {
  if-feature radius;

  description
    "Configuration of the RADIUS client.";

  list server {
    key name;
    ordered-by user;
    description
      "List of RADIUS servers used by the device.

      When the RADIUS client is invoked by a calling
      application, it sends the query to the first server in
      this list. If no response has been received within
      'timeout' seconds, the client continues with the next
      server in the list. If no response is received from any
      server, the client continues with the first server again.
      When the client has traversed the list 'attempts' times
      without receiving any response, it gives up and returns an
```

```
        error to the calling application.";

    leaf name {
        type string;
        description
            "An arbitrary name for the RADIUS server.";
    }
    choice transport {
        mandatory true;
        description
            "The transport protocol specific parameters for this
            server.";

        case udp {
            container udp {
                description
                    "Contains UDP specific configuration parameters
                    for RADIUS.";
                leaf address {
                    type inet:host;
                    mandatory true;
                    description
                        "The address of the RADIUS server.";
                }
                leaf authentication-port {
                    type inet:port-number;
                    default "1812";
                    description
                        "The port number of the RADIUS server.";
                }
                leaf shared-secret {
                    type string;
                    mandatory true;
                    nacm:default-deny-all;
                    description
                        "The shared secret which is known to both the
                        RADIUS client and server.";
                    reference
                        "RFC 2865: Remote Authentication Dial In User
                        Service";
                }
            }
        }
    }
    leaf authentication-type {
        type identityref {
            base radius-authentication-type;
        }
    }
}
```

```
        default radius-pap;
        description
            "The authentication type requested from the RADIUS
            server.";
    }
}
container options {
    description
        "RADIUS client options.";

    leaf timeout {
        type uint8 {
            range "1..max";
        }
        units "seconds";
        default "5";
        description
            "The number of seconds the device will wait for a
            response from each RADIUS server before trying with a
            different server.";
    }
    leaf attempts {
        type uint8 {
            range "1..max";
        }
        default "2";
        description
            "The number of times the device will send a query to
            all its RADIUS servers before giving up.";
    }
}

container authentication {
    nacm:default-deny-write;
    if-feature authentication;

    description
        "The authentication configuration subtree.";

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



```
    description
      "When 'radius' is used as an authentication method,
       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-authentication' 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.";
}

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.";
  }
  list ssh-key {
    key name;
    description
      "A list of public SSH keys for this user.";
    reference
      "RFC 4253: The Secure Shell (SSH) Transport Layer
       Protocol";

    leaf name {
      type string;
```

```
        description
            "An arbitrary name for the ssh key.";
    }
    leaf algorithm {
        type string;
        mandatory true;
        description
            "The public key algorithm name for this ssh key.

            Valid values are the values in the IANA Secure Shell
            (SSH) Protocol Parameters registry, Public Key
            Algorithm Names";
        reference
            "IANA Secure Shell (SSH) Protocol Parameters registry,
            Public Key Algorithm Names";
    }
    leaf key-data {
        type binary;
        mandatory true;
        description
            "The binary key data for this ssh key.";
    }
}

container system-state {
    config false;
    description
        "System group operational state.";

    container platform {
        config false;
        description
            "Contains vendor-specific information for
            identifying the system platform and operating system.";
        reference
            "IEEE Std 1003.1-2008 - sys/utsname.h";

        leaf os-name {
            type string;
            description
                "The name of the operating system in use,
                for example 'Linux'";
            reference
                "IEEE Std 1003.1-2008 - utsname.sysname";
        }
    }
}
```

```
    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
        "IEEE Std 1003.1-2008 - utsname.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
        "IEEE Std 1003.1-2008 - utsname.version";
    }
    leaf machine {
      type string;
      description
        "A vendor-specific identifier string representing
        the hardware in use.";
      reference
        "IEEE Std 1003.1-2008 - utsname.machine";
    }
  }
  container clock {
    description
      "Monitoring of the system
      date and time properties.";

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

```
rpc set-current-datetime {
  nacm:default-deny-all;
  description
    "Set the /system-state/clock/current-datetime leaf
    to the specified value.

    If the system is using NTP (i.e., /system/ntp/enabled
    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 {
      type yang:date-and-time;
      mandatory true;
      description
        "The current system date and time.";
    }
  }
}

rpc system-restart {
  nacm:default-deny-all;
  description
    "Request that the entire system be restarted immediately.
    A server SHOULD send an rpc reply to the client before
    restarting the system.";
}

rpc system-shutdown {
  nacm:default-deny-all;
  description
    "Request that the entire system be shut down immediately.
    A server SHOULD send an rpc reply to the client before
    shutting down the system.";
}

}

<CODE ENDS>
```

5. IANA Considerations

This document registers one 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 one 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

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. Authorization for access to specific portions of conceptual data and operations within this module is provided by the NETCONF access control model (NACM) [RFC6536].

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:

- o /system/clock/timezone: This choice contains the objects used to control the timezone used by the device.
- o /system/ntp: This container contains the objects used to control the Network Time Protocol servers used by the device.
- o /system/dns-resolver: This container contains the objects used to control the Domain Name System servers used by the device.
- o /system/radius: This container contains the objects used to control the Remote Authentication Dial-In User Service servers used by the device.
- o /system/authentication/user-authentication-order: This leaf controls how user login attempts are authenticated by the device.
- o /system/authentication/user: This list contains the local users enabled on the system.

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:

- o /system/platform: This container has objects which may help identify the specific NETCONF server and/or operating system implementation used on the device.
- o /system/authentication/user: This list has objects that may help identify the specific user names and password information in use

on the device.

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:

- o set-current-datetime: Changes the current date and time on the device.
- o system-restart: Reboots the device.
- o system-shutdown: Shuts down the device.

7. Change Log

-- RFC Ed.: remove this section before publication.

7.1. 00-01

- o added configuration-source identities
- o added configuration-source leaf to ntp and dns (via grouping) to choose configuration source
- o added association-type, iburst, prefer, and true leafs to the ntp-server list
- o extended the ssh keys for a user to a list of keys. support all defined key algorithms, not just dsa and rsa
- o clarified timezone-utc-offset description-stmt
- o removed '/system/ntp/server/true' leaf from data model

7.2. 01-02

- o added default-stmts to ntp-server/iburst and ntp-server/prefer leafs
- o changed timezone-location leaf to use iana-timezone typedef instead of a string

7.3. 02-03

- o removed configuration-source identities and leafs

7.4. 03-04

- o removed ndots dns resolver option
- o added radius-authentication-type identity, and identities for pap and chap, and a leaf to control which authentication type to use when communicating with the radius server
- o made 0 an invalid value for timeouts and attempts

7.5. 04-05

- o updated tree diagram explanation text

7.6. 05-06

- o changed ntp/use-ntp to ntp/enabled
- o changed ntp/ntp-server to ntp/server
- o removed /system/platform/nodename leaf
- o changed /system/name to /system/hostname
- o simplified must expression in user-authentication-order
- o added optional rounds to sha hash definition
- o clarified the crypt-hash description
- o clarified ntp descriptions
- o clarified YANG module description to indicate that some system properties are supported, not the entire system
- o clarified that system identification values are vendor specific, not the data node objects
- o clarified sec. 2.2 and 2.3 to indicate that the server should also be capable of configuring these properties
- o changed /system/dns/search from inet:host to inet:domain-name
- o changed RFC6021 reference to 6021-bis
- o changed /system/platform/nodename to /system/platform/hostname
- o changed /system/radius/server/{leafs} to be within a choice and 'udp' case statement so other transport specific parameters can augment this list or they can be added by the WG to a future version of this module. {leafs} are authentication-port and shared-secret.
- o updated YANG tree diagrams for objects added in -05 and -06

7.7. 06-07

- o updated the Abstract and Introduction
- o updated Tree diagram notation

- o identify all external servers (dns, ntp, radius) by name instead of address, in order to make the data model extensible for additional transport protocol.
- o updated the Security Considerations section with a reference to NACM.

7.8. 07-08

- o renamed the DNS transport to 'udp-and-tcp' and added references.
- o moved the operational state nodes into /system-state.

8. References

8.1. Normative References

- [FIPS.180-3.2008]
National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-3, October 2008, <http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf>.
- [I-D.ietf-netmod-iana-timezones]
Lange, J., "IANA Timezone Database YANG Module", draft-ietf-netmod-iana-timezones-00 (work in progress), July 2012.
- [I-D.ietf-netmod-rfc6021-bis]
Schoenwaelder, J., "Common YANG Data Types", draft-ietf-netmod-rfc6021-bis-03 (work in progress), June 2013.
- [IEEE-1003.1-2008]
Institute of Electrical and Electronics Engineers, "POSIX.1-2008", IEEE Standard 1003.1, March 2008.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.
- [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.
- [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.
- [RFC5607] Nelson, D. and G. Weber, "Remote Authentication Dial-In

User Service (RADIUS) Authorization for Network Access Server (NAS) Management", RFC 5607, July 2009.

- [RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, August 2010.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

8.2. Informative References

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC6557] Lear, E. and P. Eggert, "Procedures for Maintaining the Time Zone Database", BCP 175, RFC 6557, February 2012.

Authors' Addresses

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

NETMOD
Internet-Draft
Intended status: Standards Track
Expires: March 27, 2014

L. Lhotka
CZ.NIC
September 23, 2013

Modeling JSON Text with YANG
draft-lhotka-netmod-yang-json-02

Abstract

This document defines rules for presenting configuration and operational state data defined using YANG as JSON text. It does so by specifying a procedure for translating the subset of YANG-compatible XML documents to JSON text, and vice versa.

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, 2014.

Copyright Notice

Copyright (c) 2013 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	5
3. Specification of the Translation Procedure	6
3.1. Names and Namespaces	7
3.2. Mapping XML Elements to JSON Objects	9
3.2.1. The "leaf" Data Node	9
3.2.2. The "container" Data Node	9
3.2.3. The "leaf-list" Data Node	10
3.2.4. The "list" Data Node	10
3.2.5. The "anyxml" Data Node	11
3.3. Mapping YANG Datatypes to JSON Values	12
3.3.1. Numeric Datatypes	12
3.3.2. The "string" Type	12
3.3.3. The "boolean" Type	12
3.3.4. The "enumeration" Type	12
3.3.5. The "bits" Type	12
3.3.6. The "binary" Type	12
3.3.7. The "leafref" Type	13
3.3.8. The "identityref" Type	13
3.3.9. The "empty" Type	13
3.3.10. The "union" Type	13
3.3.11. The "instance-identifier" Type	14
3.4. IANA Considerations	14
3.5. Security Considerations	14
3.6. Acknowledgments	14
4. References	15
4.1. Normative References	15
4.2. Informative References	15
Appendix A. A Complete Example	16
Author's Address	19

1. Introduction

The aim of this document is define rules for presenting configuration and operational state data defined in the YANG data modeling language [RFC6020] as JavaScript Object Notation (JSON) text [JSON]. The result can be potentially applied in two different ways:

1. JSON may be used instead of the standard XML [XML] encoding in the context of the NETCONF protocol [RFC6241] and/or with existing data models expressed in YANG. An example application is the RESTCONF Protocol [RESTCONF].
2. Other documents that choose JSON to represent structured data can use YANG for defining the data model, i.e., both syntactic and semantic constraints that the data have to satisfy.

JSON mapping rules could be specified in a similar way as the XML mapping rules in [RFC6020]. This would however require solving several problems. To begin with, YANG uses XPath [XPath] quite extensively, but XPath is not defined for JSON and such a definition would be far from straightforward.

In order to avoid these technical difficulties, this document employs an alternative approach: it defines a relatively simple procedure which allows for translating the subset of XML that can be modeled using YANG to JSON, and vice versa. Consequently, validation of a JSON text against a data model can be done by translating the JSON text to XML, which is then validated according to the rules stated in [RFC6020].

The translation procedure is adapted to YANG specifics and requirements, namely:

1. The translation is driven by a concrete YANG data model and uses information about data types to achieve better results than generic XML-JSON translation procedures.
2. Various document types are supported, namely configuration data, configuration + state data, RPC input and output parameters, and notifications.
3. XML namespaces specified in the data model are mapped to namespaces of JSON objects. However, explicit namespace identifiers are rarely needed in JSON text.
4. Translation of XML attributes, mixed content, comments and processing instructions is outside the scope of this document.

Item 1 above also means that, depending on the data model, the same XML element can be translated to different JSON objects. For example,

```
<foo>123</foo>
```

is translated to

```
"foo": 123
```

if the "foo" node is defined as a leaf with the "uint8" datatype, or to

```
"foo": ["123"]
```

if the "foo" node is defined as a leaf-list with the "string" datatype, and the <foo> element has no siblings of the same name.

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 [RFC6020]:

- o anyxml
- o augment
- o container
- o data node
- o data tree
- o datatype
- o feature
- o identity
- o instance identifier
- o leaf
- o leaf-list
- o list
- o module
- o submodule

The following terms are defined in [XMLNS]:

- o local name
- o prefixed name
- o qualified name

3. Specification of the Translation Procedure

The translation procedure defines a 1-1 correspondence between the subset of YANG-compatible XML documents and JSON text. This means that the translation can be applied in both directions and is always invertible.

The translation procedure is applicable only to data hierarchies that are modelled by a YANG data model. An input XML document MAY contain enclosing elements representing NETCONF "Operations" and "Messages" layers. However, these enclosing elements are ignored by the translation procedure and do not appear in the resulting JSON document.

Any YANG-compatible XML document can be translated, except documents with mixed content. This is only a minor limitation since mixed content is marginal in YANG - it is allowed only in "anyxml" nodes.

The following sections specify rules mainly for translating XML documents to JSON text. Rules for the inverse translation are stated only where necessary, otherwise they can be easily inferred.

REQUIRED parameters of the translation procedure are:

- o YANG data model consisting of a set of YANG modules,
- o type of the input document,
- o optional features (defined via the "feature" statement) that are considered active.

The permissible types of input documents are listed in Table 1 together with the corresponding part of the data model that is used for the translation.

Document Type	Data Model Section
configuration and state data	main data tree
configuration	main data tree ("config true")
RPC input parameters	"input" nodes under "rpc"
RPC output parameters	"output" nodes under "rpc"
notification	"notification" nodes

Table 1: YANG Document Types

A particular application MAY decide to support only a subset of document types from Table 1. For instance, RESTCONF Protocol [RESTCONF] does not use notifications.

XML documents can be translated to JSON text only if they are valid instances of the YANG data model and selected document type, also taking into account the active features, if there are any.

The resulting JSON document is always a single object ([JSON], Sec. 4) whose members are translated from the original XML document using the rules specified in the following sections.

3.1. Names and Namespaces

The local part of a JSON name is always identical to the local name of the corresponding XML element.

Each JSON name lives in a namespace which is uniquely identified by the name of the YANG module where the corresponding data node is defined. If the data node is defined in a submodule, then the namespace identifier is the name of the main module to which the submodule belongs. The translation procedure MUST correctly map YANG namespace URIs to YANG module names and vice versa.

The namespace SHALL be expressed in JSON text by prefixing the local name in the following way:

```
<module name>:<local name>
```

Figure 1: Encoding a namespace identifier with a local name.

The namespace identifier MUST be used for local names that are

ambiguous, i.e., whenever the data model permits a sibling node with the same local name. Otherwise, the namespace identifier is OPTIONAL.

For example, consider the following YANG module:

```
module foomod {
  namespace "http://example.com/foomod";
  prefix "fm";
  container foo {
    leaf bar {
      type boolean;
    }
  }
}
```

If the data model consists only of this module, then the following is a valid JSON document:

```
{
  "foo": {
    "bar": true
  }
}
```

Now, assume the container "foo" is augmented from another module:

```
module barmod {
  namespace "http://example.com/barmod";
  prefix "bm";
  import foomod {
    prefix fm;
  }
  augment "/fm:foo" {
    leaf bar {
      type uint8;
    }
  }
}
```

In the data model combining "foomod" and "barmod", we have two sibling nodes with the same local name, namely "bar". In this case, a valid JSON document has to specify an explicit namespace identifier (module name) for both leaves:

```
{
  "foo": {
    "foomod:bar": true,
    "barmod:bar": 123
  }
}
```

3.2. Mapping XML Elements to JSON Objects

XML elements that are modelled as YANG data nodes are translated to a name/value pair where the name is formed from the name of the XML element using the rules in Section 3.1. The value depends on the type of the data node as specified in the following sections.

3.2.1. The "leaf" Data Node

An XML element that is modeled as YANG leaf is translated to a name/value pair and the type of the value is derived from the YANG datatype of the leaf (see Section 3.3 for the datatype mapping rules).

Example: For the leaf node definition

```
leaf foo {
  type uint8;
}
```

the XML element

```
<foo>123</foo>
```

corresponds to the JSON name/value pair

```
"foo": 123
```

3.2.2. The "container" Data Node

An XML element that is modeled as YANG container is translated to a name/object pair.

Example: For the container node definition

```
container bar {
  leaf foo {
    type uint8;
  }
}
```

the XML element

```
<bar>
  <foo>123</foo>
</bar>
```

corresponds to the JSON name/value pair

```
"bar": {
  "foo": 123
}
```

3.2.3. The "leaf-list" Data Node

A sequence of one or more sibling XML elements with the same qualified name that is modeled as YANG leaf-list is translated to a name/array pair, and the array elements are primitive values whose type depends on the datatype of the leaf-list (see Section 3.3).

Example: For the leaf-list node definition

```
leaf-list foo {
  type uint8;
}
```

the XML elements

```
<foo>123</foo>
<foo>0</foo>
```

corresponds to the JSON name/value pair

```
"foo": [123, 0]
```

3.2.4. The "list" Data Node

A sequence of one or more sibling XML elements with the same qualified name that is modeled as YANG list is translated to a name/array pair, and the array elements are JSON objects.

Unlike the XML encoding, where the list keys are required to come before any other siblings, and in the order specified by the data model, the order of members within a JSON list entry is arbitrary, because JSON objects are fundamentally unordered collections of members.

Example: For the list node definition


```
list bar {  
  key foo;  
  leaf foo {  
    type uint8;  
  }  
  leaf baz {  
    type string;  
  }  
}
```

the XML elements

```
<bar>  
  <foo>123</foo>  
  <baz>zig</baz>  
</bar>  
<bar>  
  <foo>0</foo>  
  <baz>zag</baz>  
</bar>
```

corresponds to the JSON name/value pair

```
"bar": [  
  {  
    "foo": 123,  
    "baz": "zig"  
  },  
  {  
    "foo": 0,  
    "baz": "zag"  
  }  
]
```

3.2.5. The "anyxml" Data Node

An XML element that is modeled as a YANG anyxml node is translated to a name/object pair. The content of such an element is not modelled by YANG, and there may not be a straightforward mapping to JSON text (e.g., if it is a mixed XML content). Therefore, translation of anyxml contents is necessarily application-specific and outside the scope of this document.

Example: For the anyxml node definition

```
anyxml bar;
```

the XML element

```
<bar>
  <p xmlns="http://www.w3.org/1999/xhtml">
    This is <em>very</em> cool.
  </p>
</bar>
```

may be translated to the following JSON name/value pair:

```
{
  "bar": {
    "p": "This is *very* cool."
  }
}
```

3.3. Mapping YANG Datatypes to JSON Values

3.3.1. Numeric Datatypes

A value of one of the YANG numeric datatypes ("int8", "int16", "int32", "int64", "uint8", "uint16", "uint32", "uint64" and "decimal64") is mapped to a JSON number using the same lexical representation.

3.3.2. The "string" Type

A "string" value is mapped to an identical JSON string, subject to JSON encoding rules.

3.3.3. The "boolean" Type

A "boolean" value is mapped to the corresponding JSON value 'true' or 'false'.

3.3.4. The "enumeration" Type

An "enumeration" value is mapped in the same way as a string except that the permitted values are defined by "enum" statements in YANG.

3.3.5. The "bits" Type

A "bits" value is mapped to a string identical to the lexical representation of this value in XML, i.e., space-separated names representing the individual bit values that are set.

3.3.6. The "binary" Type

A "binary" value is mapped to a JSON string identical to the lexical representation of this value in XML, i.e., base64-encoded binary

data.

3.3.7. The "leafref" Type

A "leafref" value is mapped according to the same rules as the type of the leaf being referred to.

3.3.8. The "identityref" Type

An "identityref" value is mapped to a string representing the qualified name of the identity. Its namespace MAY be expressed as shown in Figure 1. If the namespace part is not present, the namespace of the name of the JSON object containing the value is assumed.

3.3.9. The "empty" Type

An "empty" value is mapped to '[null]', i.e., an array with the 'null' value being its only element.

This representation was chosen instead of using simply 'null' in order to facilitate the use of empty leafs in common programming languages. When used in a boolean context, the '[null]' value, unlike 'null', evaluates to 'true'.

Example: For the leaf node definition

```
leaf foo {  
    type empty;  
}
```

the XML element

```
<foo/>
```

corresponds to the JSON name/value pair

```
"foo": [null]
```

3.3.10. The "union" Type

YANG "union" type represents a choice among multiple alternative types. The actual type of the XML value MUST be determined using the procedure specified in Sec. 9.12 of [RFC6020] and the mapping rules for that type are used.

For example, consider the following YANG definition:

```
leaf-list bar {  
  type union {  
    type uint16;  
    type string;  
  }  
}
```

The sequence of three XML elements

```
<bar>6378</bar>  
<bar>14.5</bar>  
<bar>infinity</bar>
```

will then be translated to this name/array pair:

```
"bar": [6378, "14.5", "infinity"]
```

3.3.11. The "instance-identifier" Type

An "instance-identifier" value is a string representing a simplified XPath specification. It is mapped to an analogical JSON string in which all occurrences of XML namespace prefixes are either removed or replaced with the corresponding module name according to the rules of Section 3.1.

When translating such a value from JSON to XML, all components of the instance-identifier MUST be given appropriate XML namespace prefixes. It is RECOMMENDED that these prefixes be those defined via the "prefix" statement in the corresponding YANG modules.

3.4. IANA Considerations

TBD.

3.5. Security Considerations

TBD.

3.6. Acknowledgments

The author wishes to thank Andy Bierman, Martin Bjorklund and Phil Shafer for their helpful comments and suggestions.

4. References

4.1. Normative References

- [JSON] Bray, T., Ed., "The JSON Data Interchange Format", draft-ietf-json-rfc4627bis-03 (work in progress), September 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for Network Configuration Protocol (NETCONF)", RFC 6020, September 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "NETCONF Configuration Protocol", RFC 6241, June 2011.
- [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>>.
- [XMLNS] Bray, T., Hollander, D., Layman, A., Tobin, R., and H. Thompson, "Namespaces in XML 1.0 (Third Edition)", World Wide Web Consortium Recommendation REC-xml-names-20091208, December 2009, <<http://www.w3.org/TR/2009/REC-xml-names-20091208>>.

4.2. Informative References

- [IF-CFG] Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-interfaces-cfg-12 (work in progress), July 2013.
- [RESTCONF] Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando, "RESTCONF Protocol", draft-bierman-netconf-restconf-01 (work in progress), September 2013.
- [XPath] Clark, J., "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>>.

Appendix A. A Complete Example

The JSON document shown below was translated from a reply to the NETCONF <get> request that can be found in Appendix D of [IF-CFG]. The data model is a combination of two YANG modules: "ietf-interfaces" and "ex-vlan" (the latter is an example module from Appendix C of [IF-CFG]). The "if-mib" feature defined in the "ietf-interfaces" module is considered to be active.

```
{
  "interfaces": {
    "interface": [
      {
        "name": "eth0",
        "type": "ethernetCsmacd",
        "enabled": false
      },
      {
        "name": "eth1",
        "type": "ethernetCsmacd",
        "enabled": true,
        "vlan-tagging": true
      },
      {
        "name": "eth1.10",
        "type": "l2vlan",
        "enabled": true,
        "base-interface": "eth1",
        "vlan-id": 10
      },
      {
        "name": "lo1",
        "type": "softwareLoopback",
        "enabled": true
      }
    ]
  },
  "interfaces-state": {
    "interface": [
      {
        "name": "eth0",
        "type": "ethernetCsmacd",
        "admin-status": "down",
        "oper-status": "down",
        "if-index": 2,
        "phys-address": "00:01:02:03:04:05",
        "statistics": {
          "discontinuity-time": "2013-04-01T03:00:00+00:00"
        }
      }
    ]
  }
}
```

```
    }
  },
  {
    "name": "eth1",
    "type": "ethernetCsmacd",
    "admin-status": "up",
    "oper-status": "up",
    "if-index": 7,
    "phys-address": "00:01:02:03:04:06",
    "higher-layer-if": [
      "eth1.10"
    ],
    "statistics": {
      "discontinuity-time": "2013-04-01T03:00:00+00:00"
    }
  },
  {
    "name": "eth1.10",
    "type": "l2vlan",
    "admin-status": "up",
    "oper-status": "up",
    "if-index": 9,
    "lower-layer-if": [
      "eth1"
    ],
    "statistics": {
      "discontinuity-time": "2013-04-01T03:00:00+00:00"
    }
  },
  {
    "name": "eth2",
    "type": "ethernetCsmacd",
    "admin-status": "down",
    "oper-status": "down",
    "if-index": 8,
    "phys-address": "00:01:02:03:04:07",
    "statistics": {
      "discontinuity-time": "2013-04-01T03:00:00+00:00"
    }
  },
  {
    "name": "lo1",
    "type": "softwareLoopback",
    "admin-status": "up",
    "oper-status": "up",
    "if-index": 1,
    "statistics": {
      "discontinuity-time": "2013-04-01T03:00:00+00:00"
    }
  }
]
```

```
    }  
  }  
}
```


Author's Address

Ladislav Lhotka
CZ.NIC

Email: lhotka@nic.cz

Internet
Internet-Draft
Intended status: Informational
Expires: April 25, 2014

D. Yeung
Y. Qu
A. Clemm
Cisco Systems
October 22, 2013

Yang Data Model for OSPF Protocol
draft-yeung-netmod-ospf-00

Abstract

This document defines a YANG data model that can be used to configure and manage OSPF.

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 25, 2014.

Copyright Notice

Copyright (c) 2013 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. Overview	2
1.1. Requirements Language	2
2. Design of Data Model	3
2.1. Overview	3
2.2. OSPFv2 and OSPFv3	4
2.3. Optional Features	4
2.4. Inheritance	4
2.5. OSPF Router Configuration	4
2.6. OSPF Address-Family Configuration	5
2.7. OSPF Area Configuration	6
2.8. OSPF Interface Configuration	7
3. OSPF Yang Module	9
4. Security Considerations	29
5. Acknowledgements	29
6. References	30
6.1. Normative References	30
6.2. Informative References	30

1. Overview

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encodings other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage OSPF. Both OSPFv2 and OSPFv3 are supported. In addition to the core OSPF protocol, features described in different separate OSPF RFC are also supported. Those non-core features are made optional in the data model provided.

1.1. Requirements Language

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].

2. Design of Data Model

Although the basis of OSPF configuration elements like routers, areas and interfaces remains the same, the detailed configuration model varies among different vendors. Differences are observed in term of how protocol engine is tied to routing domain, how multiple protocol engines could be instantiated and configuration inheritance, just to name a few.

The goal of this document is to define a data model that is capable of representing these differences. There is very little information that is designated as "mandatory", providing freedom to vendors to adapt this data model to their product implementation.

2.1. Overview

The OSPF YANG module defined in this document has all the common building blocks for OSPF protocol.

```
module: ospf
  +--rw ospf-routers
    +--rw ospf-router [version name]
      .
      .
    +--rw ospf-afs
      +--rw ospf-af [vrf-name afi safi]
        .
        .
      +--rw ospf-areas
        .
        .
      +--rw ospf-area [area-id]
        .
        .
      +--rw ospf-interfaces
        .
        .
      +--rw ospf-interface [interface]
        .
        .
```

The container ospf-routers allows specification of one or more OSPF configuration entities called ospf-router. The ospf-router is intended to match to the vendor specific OSPF configuration construct which is identified by a local identifier 'name'. The field 'version' allows support for OSPFv2 and OSPFv3.

The container `ospf-afs` includes one or more OSPF protocol engines, each encapsulated in the `ospf-af` entity. Each `ospf-af` includes information for the routing domain it is running on based on the `[vrf-name afi safi]` specification. There is no default routing domain assumed by the data model. For example, to enable OSPF on the default IPv4 routing domain of the vendor, this model requires an explicit `ospf-af` entity with the specification like `["default" "ipv4" "unicast"]`. The `ospf-af` also contains OSPF router level configuration

The `ospf-areas/ospf-area` and `ospf-intefaces/ospf-interface` contains the OSPF configuration for the area and interface level respectively

2.2. OSPFv2 and OSPFv3

The defined data model supports both OSPFv2 and OSPFv3.

The field 'version' is used to indicate the OSPF version and is a mandatory. Based on the version set, the data model change accordingly to accommodate the difference between the two versions.

2.3. Optional Features

Optional features are features beyond the basic of OSPF configurations and it is up to a vendor to decide the support of a particular feature on a particular device.

This module has declared a number of features, such as NSR, max-LSA etc.. It is intended that vendors will extend the features list.

2.4. Inheritance

This defined data model supports configuration inheritance for areas and interfaces.

Area related configurations specified in the `ospf-areas` container apply to all `ospf-area`.

Interface related configurations specified in `ospf-areas` container apply to all `ospf-interface` in all `ospf-area`'s.

Interface related configurations specified in `ospf-interfaces` container apply to all `ospf-interface` in the enclosing `ospf-area`.

2.5. OSPF Router Configuration

The container `ospf-routers` is the top level container in this data model. One or more `ospf-router` entity could be included in the `ospf-`

routers container. The main purpose of the ospf-router is to specify the OSPF version and the local name for the enclosed ospf-afs container which has the protocol configuration specifics.

```

module: ospf
  +--rw ospf-routers
    +--rw ospf-router [version name]
      +--rw version      uint8
      +--rw name         string
      .
      .

```

2.6. OSPF Address-Family Configuration

The container ospf-afs contains one or more ospf-af's each represents an OSPF protocol engine. Each ospf-af indicates the routing domain it is associated with based on [vrf-name afi safi] and contains the router level configurations.

```

module: ospf
  +--rw ospf-routers
    +--rw ospf-router [version name]
      ...
    +--rw ospf-afs
      +--rw ospf-af [vrf-name afi safi]
        +--rw vrf-name      string
        +--rw afi           enumeration
        +--rw safi          enumeration
        +--rw router-id
          | ...
        +--rw distance?      uint8
        +--rw default-originate
          | ...
        +--rw auto-cost
          | ...
        +--rw default-metric? uint32
        +--rw summary-prefix [prefix]
          | ...
        +--rw max-lsa
          | ...
        +--rw maximum
          | ...
        +--rw ignore
          | ...
        +--rw log
          | ...
        +--rw nsr

```

```

|   ...
+--rw nsf
|   ...
+--rw protocol
|   ...
+--rw spf
|   ...
+--rw snmp
|   ...
+--rw redistribute
|   ...
+--rw timers
|   ...
+--rw mpls
|   ...
+--rw fast-reroute
|   ...

```

2.7. OSPF Area Configuration

The container `ospf-areas` contains one or more `ospf-area`'s each represents an area in the OSPF protocol. Each `ospf-area` contains configurations of that area and the `ospf-interfaces` container which includes all the OSPF interfaces active in the enclosing area.

The `ospf-areas` also contains area configuration that could be inherited to all `ospf-area`'s defined. Similarly, the `ospf-area` also contains interface configuration that could be inherited to all the `ospf-interface`'s defined.

```

module: ospf
  +--rw ospf-routers
    +--rw ospf-router [version name]
      ...
    +--rw ospf-afs
      +--rw ospf-af [vrf-name afi safi]
        ...
      +--rw ospf-areas
        +--rw loopback?                               enumeration
        +--rw network-type?                             enumeration
        +--rw passive?                                   boolean
        +--rw mtu-ignore?                               boolean
        +--rw flood-reduction-enable?                   boolean
        +--rw demand-circuit?                           boolean
        +--rw demand-circuit-ignore-enable?             boolean
        +--rw prefix-suppression?                       boolean

```



```

+--rw ttl-security
...
+--rw packet-size
...
+--rw authentication
...
+--rw database-filter
...
+--rw neighbor [ip-address]
...
+--rw bfd
...
+--rw cost?                               uint16
+--rw hello-interval?                     uint16
+--rw dead-interval?                      uint16
+--rw retransmit-interval?                uint16
+--rw transmit-delay?                    uint16
+--rw ospf-area [area-id]
    +--rw area-id                          uint32
    +--rw area-notation?                  enumeration
    +--rw (area-type)?
        ...
    +--rw default-cost?                    uint32
    +--rw range [prefix]
        ...
    +--rw virtual-link
        ...
    +--rw sham-link
        ...
+--rw mpls-te-config
...

```

2.8. OSPF Interface Configuration

The container `ospf-areas` contains one or more `ospf-interface`'s each represents an interface in the OSPF protocol. Each `ospf-interface` contains configurations of that interface.

The `ospf-interfaces` also contain interface configuration that could be inherited to all `ospf-interface`'s defined.

```

module: ospf
  +--rw ospf-routers
    +--rw ospf-router [version name]
      ...
    +--rw ospf-afs
      +--rw ospf-af [vrf-name afi safi]

```

```

...
+--rw ospf-areas
    ...
    +--rw ospf-interfaces
        +--rw network-type?          enumeration
        +--rw passive?               boolean
        +--rw mtu-ignore?            boolean
        +--rw flood-reduction-enable? boolean
        +--rw demand-circuit?       boolean
        +--rw demand-circuit-ignore-enable? boolean
        +--rw prefix-suppression?   boolean
        +--rw ttl-security
            ...
        +--rw packet-size
            ...
        +--rw authentication
            ...
        +--rw database-filter
            ...
        +--rw neighbor [ip-address]
            ...
        +--rw bfd
            ...
        +--rw cost?                  uint16
        +--rw hello-interval?       uint16
        +--rw dead-interval?        uint16
        +--rw retransmit-interval?   uint16
        +--rw transmit-delay?       uint16
        +--rw ospf-interface [interface]
            +--rw interface          if:interface-ref
            +--rw network-type?      enumeration
            +--rw passive?           boolean
            +--rw mtu-ignore?        boolean
            +--rw flood-reduction-enable? boolean
            +--rw demand-circuit?    boolean
            +--rw demand-circuit-ignore-enable? boolean
            +--rw prefix-suppression? boolean
            +--rw ttl-security
                ...
            +--rw packet-size
                ...
            +--rw authentication
                ...
            +--rw database-filter
                ...
            +--rw neighbor [ip-address]
                ...
            +--rw bfd

```

```
...
+--rw cost?                uint16
+--rw hello-interval?      uint16
+--rw dead-interval?       uint16
+--rw retransmit-interval? uint16
+--rw transmit-delay?      uint16
```

3. OSPF Yang Module

```
<CODE BEGINS>
module ospf {
  namespace "urn:cisco:params:xml:ns:yang:ospf";
  // replace with IANA namespace when assigned
  prefix ospf;

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-interfaces {
    prefix if;
  }

  organization
    "Cisco Systems
     170 West Tasman Drive
     San Jose, CA 95134-1706
     USA";
  contact
    "Derek Yeung myeung@cisco.com
     Yingzhen Qu yiqu@cisco.com
     Alexander Clemm alex@cisco.com";

  description
    "This YANG module defines the generic configuration
     data for OSPF, which is common across all of the vendor
     implementations of the protocol. It is intended that the module
     will be extended by vendors to define vendor-specific
     OSPF configuration parameters and policies,
     for example route maps or route policies.

     Terms and Acronyms

     OSPF (ospf): Open Shortest Path First

     IP (ip): Internet Protocol
```

```
    IPv4 (ipv4):Internet Protocol Version 4

    IPv6 (ipv6): Internet Protocol Version 6

    MTU (mtu) Maximum Transmission Unit
    ";

revision 2013-10-20 {
    description
        "Initial revision.";
}

feature flood-reduction {
    description
        "OSPF Flood Reduction.";
}

feature demand-circuit-ignore {
    description
        "Ignore demand circuit auto-negotiation requests.";
}

feature security {
    description
        "Enable/Disable TTL security.";
}

feature packet-size {
    description
        "Customize size of OSPF packets up to MTU.";
}

feature nsf-cisco {
    description
        "Enable Cisco Non Stop Forwarding.";
}

feature loopback-config {
    description
        "OSPF loopback configuration.";
}

feature max-lsa {
    description
        "Maximum number of LSAs OSPF process will receive.";
}

feature maximum-ecmp {
```

```
    description
      "Maximum number of ECMP paths.";
  }

  feature nsr {
    description
      "Enable NSR.";
  }

  feature spf {
    description
      "SPF configuration";
  }

  feature nssa-only {
    description
      "Limit summary to NSSA areas.";
  }

  feature nssa-ext-capability {
    description
      "Send domain specific capabilities into NSSA.";
  }

  grouping interface-internal-inherit-config {
    leaf cost {
      description
        "Interface cost.";
      type uint16 {
        range "1..65535";
      }
    }

    leaf hello-interval {
      description
        "Time between HELLO packets.";
      units seconds;
      type uint16 {
        range "1..65535";
      }
    }

    leaf dead-interval {
      description
        "Interval after which a neighbor is declared dead.";
      units seconds;
      type uint16 {
        range "1..65535";
      }
    }
  }
```

```
    }
    must "dead-interval > ../hello-interval" {
        error-app-tag dead-interval-invalid;
        error-message "The dead interval must be "
            + "larger than the hello interval";
    }
}

leaf retransmit-interval {
    description
        "Time between retransmitting lost link state
        advertisements.";
    units seconds;
    type uint16 {
        range "1..65535";
    }
}

leaf transmit-delay {
    description
        "Estimated time needed to send link-state update.";
    units seconds;
    type uint16 {
        range "1..65535";
    }
}
} // interface-internal-inherit-config

grouping interface-inherit-config {
    leaf network-type {
        description
            "Network type.";
        type enumeration {
            enum "broadcast" {
                description
                    "Specify OSPF broadcast multi-access network.";
            }
            enum "non-broadcast" {
                description
                    "Specify OSPF NBMA network.";
            }
            enum "point-to-multipoint" {
                description
                    "Specify OSPF point-to-multipoint network.";
            }
            enum "point-to-point" {
                description
                    "Specify OSPF point-to-point network.";
            }
        }
    }
}
```

```
    }  
  }  
}  
  
leaf passive {  
  description  
    "Enable/Disable passive.";  
  type boolean;  
}  
  
leaf mtu-ignore {  
  description  
    "Enable/Disable ignoring of MTU in DBD packets.";  
  type boolean;  
}  
  
leaf flood-reduction-enable {  
  if-feature flood-reduction;  
  type boolean;  
}  
  
leaf demand-circuit {  
  description  
    "Enable/Disable demand circuits.";  
  type boolean;  
}  
  
leaf demand-circuit-ignore-enable {  
  if-feature demand-circuit-ignore;  
  type boolean;  
}  
  
leaf prefix-suppression {  
  description  
    "Suppress advertisement of the prefixes.";  
  type boolean;  
}  
  
container ttl-security {  
  if-feature security;  
  leaf enable {  
    type boolean;  
  }  
  leaf hops {  
    type uint8 {  
      range "1..254";  
    }  
  }  
}
```

```
    }  
  
    container packet-size {  
      if-feature packet-size;  
      description  
        "Customize size of OSPF packets up to MTU.";  
      leaf enable {  
        type boolean;  
      }  
      leaf size {  
        type uint16 {  
          range "576..10000";  
        }  
      }  
    }  
  }  
}
```

```
container authentication {  
  when "../../version = 2";  
  description "Configure authentication.";  
  leaf enable {  
    type boolean;  
  }  
  choice authentication-type-selection {  
    case clear-text {  
      description  
        "Use clear-text authentication.";  
      leaf key {  
        type string {  
          length "1..8";  
        }  
      }  
    }  
    case message-digest {  
      description  
        "Use message-digest authentication.";  
      list message-digest-key {  
        key key-id;  
        leaf key-id {  
          type uint8 {  
            range "1..255";  
          }  
        }  
        leaf algorithm {  
          type enumeration {  
            enum "md5";  
          }  
        }  
        leaf key {
```



```
        type string {
            length "1..8";
        }
    }
}

container database-filter {
    description
        "Filter OSPF LSA during synchronization and flooding";
    leaf enable {
        type boolean;
    }
    leaf lsa-type {
        type enumeration {
            enum "all";
        }
    }
    leaf direction {
        type enumeration {
            enum "out";
        }
    }
}

list neighbor {
    description
        "Specify a neighbor router.";
    key ip-address;
    leaf ip-address {
        type inet:ip-address;
    }
    leaf cost {
        type uint16 {
            range "1..65535";
        }
    }
    leaf poll-interval {
        units seconds;
        type uint16 {
            range "1..65535";
        }
    }
    leaf priority {
        type uint8 {
            range "1..255";
        }
    }
}
```

```
    }  
  }  
}  
  
container bfd {  
  leaf enable {  
    description  
      "enable bfd.";  
    type boolean;  
  }  
  
  leaf minimum-interval {  
    description  
      "bfd hello interval";  
    type uint16 {  
      range "15..30000";  
    }  
  }  
  
  leaf multiplier {  
    description  
      "detect multiplier";  
    type uint8 {  
      range "2..50";  
    }  
  }  
}  
  
uses interface-internal-inherit-config;  
  
} // grouping interface-inherit-config  
  
grouping area-inherit-config {  
  leaf loopback {  
    if-feature loopback-config;  
    description  
      "OSPF loopback configuration.";  
    type enumeration {  
      enum "stub";  
      enum "host";  
    }  
  }  
}  
  
} // grouping area-inherit-config  
  
container ospf-routers {  
  list ospf-router {  
    description  
      "This is a top-level container for the OSPF router.";  
  }  
}
```

```
key "version name";
leaf version {
  description
    "OSPF version.";
  mandatory true;
  type uint8 {
    range "2..3";
  }
}

leaf name {
  description
    "Name, combined with
    ospf-routers/ospf_router/ospf-afs/ospf-af,
    identify an OSPF protocol instance.";
  mandatory true;
  type string;
}

container ospf-afs {
  list ospf-af {
    key "vrf-name afi safi";
    leaf vrf-name {
      type string;
    }
    leaf afi {
      type enumeration {
        enum "ipv4" {
          description
            "IPv4";
        }
        enum "ipv6" {
          description
            "IPv6";
        }
      }
    }
    leaf safi {
      type enumeration {
        enum "unicast" {
          description
            "unicast";
        }
      }
    }
  }

  container router-id {
    description
```

```
        "Specify the router-id for this OSPF process.";
    leaf enable {
        type boolean;
    }
    choice config-type {
        case static {
            leaf ip-address {
                type boolean;
            }
        }
        case auto-config {
            leaf enable-auto-config {
                type boolean;
            }
        }
    }
}

leaf distance {
    description
        "Define an administrative distance.";
    type uint8 {
        range "1..255";
    }
}

container default-originate {
    description
        "Control distribution of default route information.";
    leaf enable {
        type boolean;
        default "false";
    }
}

container auto-cost {
    description
        "Calculate OSPF interface cost according to
        bandwidth.";
    leaf enable {
        type boolean;
    }
    leaf reference-bandwidth {
        type uint32 {
            range "1..4294967";
        }
    }
}
```

```
leaf default-metric {
  description
    "Calculate OSPF interface cost according to
    bandwidth.";
  type uint32 {
    range "1..16777214";
  }
}

list summary-prefix {
  description
    "Configure IP address summaries";
  key "prefix";
  leaf prefix {
    type inet:ip-prefix;
  }
  leaf advertise {
    type boolean;
  }
  leaf tag {
    type uint32 {
      range "0..4294967295";
    }
  }
  leaf nssa-only-enable {
    if-feature nssa-only;
    type boolean;
  }
}

container max-lsa {
  if-feature max-lsa;
  description
    "Maximum number of LSAs OSPF process will
    receive.";
  leaf max {
    description
      "Maximum number of non self-generated LSAs
      this process can receive.";
    type uint32 {
      range "1..4294967294";
    }
  }
  leaf threshold {
    description
      "Threshold value (%) at which to generate
      a warning msg.";
    type uint8 {
```

```
        range "1..100";
    }
}
leaf warning-only {
    description
        "Only give warning message when limit is
        exceeded.";
    type boolean;
}
leaf ignore-time {
    description
        "Number of minutes during which all adjacencies
        are suppressed.";
    type uint16 {
        range "1..17895";
    }
}
leaf ignore-count {
    description
        "Count on how many times adjacencies can be
        suppressed.";
    type uint16 {
        range "1..65534";
    }
}
leaf reset-time {
    description
        "number of minutes after which ignore-count is
        reset to zero.";
    type uint16 {
        range "2..35791";
    }
}
}

container maximum {
    if-feature maximum-ecmp;
    description
        "Set OSPF limits.";
    leaf paths {
        description
            "Maximum number of ECMP paths.";
        type uint16 {
            range "1..32";
        }
    }
    leaf interfaces {
        description
```

```
        "Maximum number of interfaces.";
    type uint16 {
        range "1..1024";
    }
}
container redistributed-prefixes {
    description
        "Maximum number of prefixes redistributed.";
    leaf max {
        type uint32 {
            range "1..4294967295";
        }
    }
    leaf threshold {
        description
            "Threshold value (%) at which to generate
            a warning msg.";
        type uint8 {
            range "1..100";
        }
    }
    leaf warning-only {
        description
            "Only give warning message when limit is
            exceeded.";
        type boolean;
    }
}

container ignore {
    description
        "Do not complain about specific event.";
    leaf-list lsa {
        type enumeration {
            enum "mospf";
        }
    }
}

container log {
    description
        "Log ospf info.";
    leaf adjacency-changes {
        type boolean;
    }
}
```

```
    container nsr {
      if-feature nsr;
      description
        "Enable NSR.";
      leaf enable {
        type boolean;
      }
    }

    container nsf {
      choice nsf-type {
        case cisco {
          if-feature nsf-cisco;
          description
            "Enable Cisco Non Stop Forwarding.";
          leaf cisco-enable {
            type boolean;
          }
        }
        case ietf {
          description
            "Enable Cisco Non Stop Forwarding.";
          leaf ietf-enable {
            type boolean;
          }
          leaf ietf-helper-enable {
            type boolean;
          }
        }
      }
    }

    container protocol {
      description
        "Protocol specific configuration.";
      leaf shutdown {
        type boolean;
      }
    }

    container spf {
      if-feature spf;
      description
        "SPF configuration";
      leaf prefix-priority-policy {
        type string;
      }
    }
  }
```



```
    container snmp {
      description
        "Adjust ospf snmp parameters";
      leaf context {
        type string;
      }
    }

    container redistribute {
      description
        "Redistribute information from another
        routing protocol.
        This grouping is intended to be augmented by
        vendors to implement vendor-specific protocol
        redistribution configuration options.";
      choice protocol {
        case bgp {
          leaf enable-bgp {
            type boolean;
          }
        }
        case ospf {
          leaf enable-ospf {
            type boolean;
          }
        }
        case isis {
          leaf enable-isis {
            type boolean;
          }
        }
        case connected {
          leaf enable-connected {
            type boolean;
          }
        }
        case eigrp {
          leaf enable-eigrp {
            type boolean;
          }
        }
        case mobile {
          leaf enable-mobile {
            type boolean;
          }
        }
        case static {
          leaf enable-static {
```

```
        type boolean;
    }
}
case rip {
    leaf enable-rip {
        type boolean;
    }
}
}
}

container timers {
    description
        "Adjust routing timers.
        This grouping is intended to be augmented
        by vendors to implement vendor-specific
        protocol timers configuration options";

    container lsa {
        leaf min-arrival {
            description
                "The minimum interval in milliseconds
                between accepting the same";
            units milliseconds;
            type uint32 {
                range "0..600000";
            }
        }
        leaf refresh {
            description
                "The minimum interval in seconds
                between refresh";
            units seconds;
            type uint32 {
                range "1800..2700";
            }
        }
    }
}

container throttle-lsa {
    leaf delay {
        description
            "Delay to generate first occurrence of
            LSA in milliseconds";
        units milliseconds;
        type uint32 {
            range "0..600000";
        }
    }
}
```

```
leaf min-delay {
  description
    "The Minimum delay between originating
    the same LSA in milliseconds";
  units milliseconds;
  type uint32 {
    range "0..600000";
  }
}
leaf max-delay {
  description
    "The Maximum delay between originating
    the same LSA in milliseconds";
  units milliseconds;
  type uint32 {
    range "0..600000";
  }
}
}

container throttle-spf {
  leaf delay {
    description
      "Delay between receiving a change to
      SPF calculation in milliseconds";
    units milliseconds;
    type uint32 {
      range "0..600000";
    }
  }
  leaf min-delay {
    description
      "Delay between first and second SPF
      calculation in milliseconds";
    units milliseconds;
    type uint32 {
      range "0..600000";
    }
  }
  leaf max-delay {
    description
      "Maximum wait time in milliseconds
      for SPF calculations";
    units milliseconds;
    type uint32 {
      range "0..600000";
    }
  }
}
```

```
    }  
  }  
  
  container mpls {  
    description  
      "OSPF MPLS configuraions.";  
    leaf te-rid {  
      description  
        "Traffic Engineering stable  
         IP address for system.";  
      type if:interface-ref;  
    }  
    leaf ldp-sync-enable {  
      description  
        "Enable LDP IGP synchronization.";  
      type boolean;  
    }  
    leaf ldp-autoconfig-enable {  
      description  
        "Enable LDP IGP interface  
         auto-configuration.";  
      type boolean;  
    }  
  }  
}  
  
container fast-reroute {  
  leaf fast-reroute-enable {  
    description  
      "IP fast reroute.";  
    type boolean;  
  }  
}  
  
container ospf-areas {  
  description  
    "The top level container for the list  
     of areas of the OSPF router.";  
  
  uses area-inherit-config;  
  uses interface-inherit-config;  
  
  list ospf-area {  
    key "area-id";  
    leaf area-id {  
      type uint32;  
    }  
  
    leaf area-notation {
```

```
description
  "Sets the default area notation.";
type enumeration {
  enum "decimal";
  enum "dot";
}
}
choice area-type {
  case nssa {
    description
      "Specify area as a NSSA area.";
    leaf redistribute {
      type boolean;
    }
    leaf nssa-summary {
      type boolean;
    }
    leaf nssa-ext-capability {
      if-feature nssa-ext-capability;
      type boolean;
    }
  }
  container default-information-originate {
    description
      "Originate Type 7 default into NSSA area.";
    leaf metric {
      type uint16 {
        range "1..65535";
      }
    }
    leaf metric-type {
      type uint8 {
        range "1..2";
      }
    }
  }
}
}
case stub {
  description
    "Specify area as a stub area.";
  leaf stub-summary {
    type boolean;
  }
  leaf stub-ext-capability {
    type boolean;
  }
}
}
```

```
leaf default-cost {
  description
    "Set the summary default-cost of a
    NSSA/stub area.";
  type uint32 {
    range "1..16777215";
  }
}

list range {
  description
    "Summarize routes matching
    address/mask (border routers only)";
  key "prefix";
  leaf prefix {
    type inet:ip-prefix;
  }
  leaf advertise {
    type boolean;
  }
  leaf cost {
    type uint32 {
      range "0..16777214";
    }
  }
}

container virtual-link {
  description
    "Define a virtual link";
  leaf router-id {
    type inet:ip-address;
  }

  uses interface-internal-inherit-config;
}

container sham-link {
  leaf local-id {
    description
      "Address of the local end-point.";
    type inet:ip-address;
  }
  leaf remote-id {
    description
      "Address of the remote end-point.";
    type inet:ip-address;
  }
}
```

```
        uses interface-internal-inherit-config;
    }

    container mpls-te-config {
        leaf mpls-te-enable {
            description
                "Enable an ospf area to run MPLS
                Traffic Engineering.";
            type boolean;
        }
    }

    container ospf-interfaces {
        description
            "The top level container for the
            list of interfaces of the OSPF router.";

        uses interface-inherit-config;

        list ospf-interface {
            key "interface";
            leaf interface {
                type if:interface-ref;
            }
            uses interface-inherit-config;
        } // list ospf-interface
    } // container ospf-interfaces
} // list ospf-area
} // container ospf-areas
} // container ospf-af
} // container ospf-afs
} // list ospf-router
} // container ospf-routers
}
<CODE ENDS>
```

4. Security Considerations

The data model defined does not create any security implications.

This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

5. Acknowledgements

The authors would like to thank Gaurav Gupta and many others for review and comments.

This document was produced using Marshall Rose's xml2rfc tool.

6. References

6.1. 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.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

6.2. Informative References

- [I-D.ietf-netmod-interfaces-cfg]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-interfaces-cfg-12 (work in progress), July 2013.
- [I-D.ietf-netmod-routing-cfg]
Lhotka, L., "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-11 (work in progress), October 2013.

Authors' Addresses

Derek Yeung
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

EMail: myeung@cisco.com

Yingzhen Qu
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

EMail: yiqu@cisco.com

Alexander Clemm
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

EMail: alex@cisco.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 16, 2014

A. Zhdankin
K. Patel
A. Clemm
Cisco
July 15, 2013

Yang Data Model for BGP Protocol
draft-zhdankin-netmod-bgp-cfg-00.txt

Abstract

This document defines a YANG data model that can be used to configure and manage BGP.

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 16, 2014.

Copyright Notice

Copyright (c) 2013 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	2
1.1. Requirements Language	3
2. Definitions and Acronyms	3
3. The Design of the Core Routing Data Model	4
3.1. Overview	4
3.2. BGP Router Configuration	4
3.2.1. AF Configuration	5
3.2.1.1. AF Specific Protocol Configuration	7
3.2.1.2. BGP Bestpath Configuration	7
3.2.1.3. BGP Neighbor Configuration	8
3.2.1.4. BGP Dampening	8
3.2.1.5. BGP Route Aggregation	8
3.2.1.6. BGP Redistribution	8
3.2.2. BGP Neighbor Configuration	8
3.2.3. BGP RPKI	10
3.3. Prefix Lists	10
4. BGP Yang Module	11
5. IANA Considerations	38
6. Security Considerations	38
7. Acknowledgements	38
8. References	38
8.1. Normative References	38
8.2. Informative References	39
Authors' Addresses	39

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encodings other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage BGP. The data model is very comprehensive in scope, resulting in a very large module being defined. When contemplating whether it would be appropriate to introduce a data model of such a large scope, we decided that there would be value in particular because BGP defines such a rich set of features, which makes the problem arising from heterogeneity involved when managing these features quite pronounced. Also, there is very little information that is designated as "mandatory", leaving the decision which capabilities to actually support to product implementations.

There are several distinct parts of the data model. The first part, by far the largest, serves to configure and manage BGP itself. It defines a large set of control knobs for that purpose, as well as a few data nodes that can be used to monitor health and gather statistics. The second part, much smaller than the first, defines a data model for the configuration of AS-Path and prefix-based filter lists, in essence policies that define the exchange of BGP messages between BGP peers. Together they form a complete data model that serves as a framework for configuration and management of BGP protocol and its policies.

The YANG module defined in this document has all the common building blocks for BGP protocol namely: Neighbor List, Address Family specific Parameters, Protocol Bestpath specific Parameters, Prefix based Filter Lists, and AS-PATH based Filter Lists.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. Definitions and Acronyms

AF: Address Family

AS: Autonomous System

BGP: Border Gateway Protocol

HTTP: Hyper-Text Transfer Protocol

JSON: JavaScript Object Notation

L2VPN: Layer 2 VPN

NETCONF: Network Configuration Protocol

NSAP: Network Service Access Point

ReST: Representational State Transfer, a style of stateless interface and protocol that is generally carried over HTTP

RPKI: Resource Public Key Infrastructure

RTFilter: Route Filter

VPN: Virtual Private Network

YANG: A data definition language for NETCONF

3. The Design of the Core Routing Data Model

3.1. Overview

The overall data model consists of two main components, each contained in its own separate container. Container "bgp-router" is used to configure and manage BGP itself. It is by far the largest part of the model. Container "prefix-lists" is used to configure BGP prefix lists, defining the rules and policies as which BGP information to share with which other nodes.

3.2. BGP Router Configuration

The overall structure of the "bgp-router" part of the model is depicted in the following diagram. Brackets enclose list keys, "rw" means configuration data, "?" designates optional nodes. The figure does not depict all definitions; it is intended to illustrate the overall structure.

```
module: bgp
  +--rw bgp-router
  |   +--rw local-as-number?      uint32
  |   +--rw local-as-identifier?  inet:ip-address
  |   +--rw rpki-config
  |   |
  |   | .....
  |   |
```

```

|   +--rw af-configuration
|   |   .....
+--rw bgp-neighbors
|   .....

```

The key components of the "bgp-router" model concern the configuration of the BGP neighbors, of the Resource Public Key Infrastructure (RPKI), and of address families (AF). Each is defined in the following subsections.

3.2.1. AF Configuration

AF-configuration is used to configure and manage BGP configuration on an address family basis. BGP is designed to carry routing information for multiple different address families as specified in [RFC4760]. AF-Configuration is indexed by (router-AS, AFI, SAFI, VRFID) [RFC4760] and [RFC4364]. It contains any AF specific protocol configuration, BGP Bestpath configuration parameters, BGP neighbor configuration parameters, BGP dampening parameters, BGP route aggregation parameters, and any BGP policy configuration like redistribution.

The overall structure of the AF Configuration data model is depicted in the following diagram. As before, brackets enclose list keys, "rw" means configuration data, "?" designates optional nodes, parantheses indicate choices. The figure does not depict all definitions; it is intended to illustrate the overall model structure. Roughly speaking, address family configuration allows for separate configuration of IPv4, IPv6, L2VPN, NSAP, VPNv4 and VPNv6 address families, as well as route filters. Within each address family, you have additional substructure, for example, to distinguish between configuration of unicast and multicast.

```

module: bgp
+--rw bgp-router
|   .....
|   +--rw af-configuration
|   |   +--rw ipv4
|   |   |   +--rw mdt
|   |   |   |   .....
|   |   |   +--rw multicast
|   |   |   |   +--rw bgp
|   |   |   |   |   .....
|   |   |   |   +--rw auto-summary?      boolean
|   |   |   |   +--rw aggregate-address?  inet:ip-address
|   |   |   |   +--rw distance?           uint8
|   |   |   |   +--rw network?            inet:ip-address

```

```

+--rw (protocol)?
|   .....
+--rw default-metric?      uint32
+--rw unicast
+--rw bgp
|   .....
+--rw auto-summary?       boolean
+--rw aggregate-address?   inet:ip-address
+--rw distance?           uint8
+--rw network?            inet:ip-address
+--rw (protocol)?
|   .....
+--rw number-of-path?      uint8
+--rw ibgp-number-of-path? uint8
+--rw synchronization?     boolean
+--rw mvpn
+--rw bgp
|   .....
+--rw auto-summary?       boolean
+--rw ipv6
+--rw multicast
+--rw bgp
|   .....
+--rw aggregate-address?   inet:ip-address
+--rw distance?           uint8
+--rw network?            inet:ip-address
+--rw (protocol)?
|   .....
+--rw unicast
+--rw bgp
|   .....
+--rw aggregate-address?   inet:ip-address
+--rw distance?           uint8
+--rw network?            inet:ip-address
+--rw (protocol)?
|   .....
+--rw default-metric?      uint32
+--rw number-of-path?      uint8
+--rw ibgp-number-of-path? uint8
+--rw synchronization?     boolean
+--rw mvpn
|   .....
+--rw l2vpn
+--rw vpls
|   .....
+--rw nsap
+--rw unicast
+--rw bgp

```

```

|         ....
|         +--rw default-metric?           uint32
|         +--rw number-of-path?           uint8
|         +--rw ibgp-number-of-path?     uint8
|         +--rw network?                  inet:ip-address
|         +--rw (protocol)?
|         |         ....
|         +--rw synchronization?         boolean
+--rw rtfilter
|   +--rw unicast
|   |   ....
+--rw vpnv4
|   +--rw unicast
|   |   +--rw bgp
|   |   |   ....
|   |   +--rw number-of-path?           uint8
|   |   +--rw ibgp-number-of-path?     uint8
+--rw multicast
|   +--rw bgp
|   |   ....
|   +--rw number-of-path?               uint8
|   +--rw ibgp-number-of-path?         uint8
+--rw vpnv6
|   +--rw unicast
|   |   +--rw bgp
|   |   |   ....

```

The key AF configuration components are described in the following subsections.

3.2.1.1. AF Specific Protocol Configuration

AF specific protocol configuration involves configuration of the parameters that are specific to a given AF. For instance, configuration parameters specific to the consistency checking between prefixes and labels are specific to address families that are enabled with Labels. Similarly redistribution of routes from other protocols is specific to Address Families that are supported in other protocols.

3.2.1.2. BGP Bestpath Configuration

BGP BestPath Configuration Parameters involves configuration of the parameters that influence the BGP Bestpath decision. For instance, the ignore-as-path command allows BGP process to ignore as-path length check. The ignore-routerid command allows BGP process to ignore routerid check. The ignore-igp-metric command allows BGP

process to ignore igp metric check. The ignore-cost-community command allows BGP process to ignore cost communities. The MED related commands influence MED comparison in the BGP Bestpath decision.

3.2.1.3. BGP Neighbor Configuration

BGP Neighbor Configuration Parameters involves configuration of the parameters that are neighbor address family specific. These commands include neighbor capabilities, neighbor policies and any protocol related parameters that are specific to BGP neighbor.

3.2.1.4. BGP Dampening

BGP Dampening Parameters involves configuration of the parameters that influence BGP Route Dampening. These parameters allow enabling of Route Dampening on an address family level. The Dampening configuration also allows configuration of Dampening specific parameters like max suppress time, reuse threshold, half life, and the suppress threshold.

3.2.1.5. BGP Route Aggregation

BGP Route Aggregation Parameters involves configuration of the parameters that enables BGP Route Aggregation.

3.2.1.6. BGP Redistribution

BGP Route Redistribution Parameters involves configuration of the parameters that enables BGP Route Redistribution from and to the BGP protocol.

3.2.2. BGP Neighbor Configuration

Bgp-neighbor is used to configure and manage BGP neighbors. BGP neighbor configuration is indexed by af-configuration, neighbor address and neighbor-AS. It contains configuration for any policies that are configured for a neighbor on an inbound or an outbound, any transport related configuration parameters, any protocol related configuration parameters, and any protocol capabilities related configuration parameters.

The following diagram depicts the overall structure of the BGP Neighbors subtree. Brackets enclose list keys, "rw" means configuration, "ro" operational state data, and "?" designates optional nodes. Parentheses enclose choice and case nodes. The figure does not depict all definitions; it is intended to illustrate the overall structure.

```

module: bgp
+ ....
+--rw bgp-neighbors
|   +--rw bgp-neighbor [as-number]
|       +--rw as-number                               uint32
|       +--rw (peer-address-type)?
|           |
|           | .....
|       +--rw prefix-list?                             prefix-list-ref
|       +--rw default-action?                         actions-enum
|       +--rw af-specific-config
|           +--rw ipv4
|               +--rw mdt
|                   |
|                   | .....
|               +--rw unicast
|                   |
|                   | .....
|               +--rw multicast
|                   |
|                   | .....
|               +--rw mvpn
|                   |
|                   | .....
|           +--rw ipv6
|               +--rw unicast
|                   |
|                   | .....
|               +--rw multicast
|                   |
|                   | .....
|               +--rw mvpn
|                   |
|                   | .....
|           +--rw l2vpn
|               +--rw evpn
|                   |
|                   | .....
|               +--rw vpls
|                   |
|                   | .....
|           +--rw nsap
|               +--rw unicast
|                   |
|                   | .....
|           +--rw rtfilter
|               +--rw unicast
|                   |
|                   | .....
|           +--rw vpnv4
|               +--rw unicast
|                   |
|                   | .....
|               +--rw multicast
|                   |
|                   | .....
|           +--rw vpnv6
|               +--rw unicast
|                   |
|                   | .....
|               +--rw multicast
|                   |
|                   | .....
+--rw bgp-neighbor-state

```

```
|      .....  
|--rw bgp-neighbor-statistics  
|      .....  
|
```

3.2.3. BGP RPKI

rpki-config is used to configure and manage BGP Origin Validation. This feature is specific to IPv4 and IPv6 Address Families. It is indexed by af-configuration. It contains the configuration commands for the BGP RPKI Server, RPKI RTR Protocol and the BGP protocol. This includes configuration for the Server address, Server preference, RPKI RTR protocol specific parameters, choice of a transport for RPKI RTR Protocol, and BGP specific parameters including enabling and disabling of this feature for IBGP and EBGP routes.

The structure of the RPKI configuration data model is depicted below, per the same conventions used in the earlier diagrams.

```

module: bgp
|--rw bgp-router
|
|   ....
|   |--rw rpki-config
|   |   |--rw cache-server-config
|   |   |   ....
|   |   |--rw validation-config
|   |   |   ....
|   |   |--rw bestpath-computation
|   |   |   ....

```

3.3. Prefix Lists

BGP Prefix Lists are used to manipulate Prefix information carried within a BGP. The prefix information carried within BGP is filtered or allowed using BGP Prefix Lists. BGP Prefix Lists consists of an ordered set of one or more rules that describe IPv4 or IPv6 prefixes range and an associated action rule that describes whether the matching prefixes should be dropped or permitted. The Prefix Lists are usually applied to a BGP neighbor as part of an inbound policy (applied to prefixes received by a neighbor) or an outbound policy (applied to prefixes sent by a neighbor).

The structure of the prefix list configuration data model is depicted below, per the same conventions used in the earlier diagrams.

```

module: bgp
.....
+--rw prefix-lists
  +--rw prefix-list [prefix-list-name]
    +--rw prefix-list-name    string
    +--rw prefixes
      +--rw prefix [seq-nr]
        +--rw seq-nr          uint16
        +--rw prefix-filter
          +--rw (ip-address-group)?
          | .....
          +--rw action          actions-enum
          +--rw statistics
            .....

```

Prefix lists are defined in a list in a designated container. Each prefix list in turn contains a list of prefixes, indexed by a sequence number. Each prefix is comprised of a prefix filter, used to match BGP packets, an action that is applied when a filter matches, and a set of statistics that indicate how often individual prefixes are applied.

4. BGP Yang Module

<CODE BEGINS> file "bgp@2013-06-25.yang"

```

module bgp {
  namespace "urn:cisco:params:xml:ns:yang:bgp";
  // replace with IANA namespace when assigned
  prefix bgp;

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

  organization
    "Cisco Systems
     170 West Tasman Drive
     San Jose, CA 95134-1706
     USA";
  contact
    "Aleksandr Zhdankin azhdanki@cisco.com
     Keyur Patel keyupate@cisco.com

```

Alexander Clemm alex@cisco.com";

description

"This YANG module defines the generic configuration data for BGP, which is common across all of the vendor implementations of the protocol. It is intended that the module will be extended by vendors to define vendor-specific BGP configuration parameters and policies, for example route maps or route policies.

Terms and Acronyms

BGP (bgp): Border Gateway Protocol

IP (ip): Internet Protocol

IPv4 (ipv4): Internet Protocol Version 4

IPv6 (ipv6): Internet Protocol Version 6

MED(med): Multi Exit Discriminator

IGP (igp): Interior Gateway Protocol

MTU (mtu) Maximum Transmission Unit
";

revision 2013-06-01 {

description

"Initial revision.";

}

typedef prefix-list-ref {

description

"A reference to the prefix list which a bgp-neighbor can use.";

type leafref {

path "/prefix-lists/prefix-list/prefix-list-name";

}

}

typedef neighbour-ref {

description

"A reference to the bgp-neighbor.";

type leafref {

path "/bgp-neighbors/bgp-neighbor/as-number";

}

}

```
typedef bgp-peer-admin-status {
  description
    "Administrative status of a BGP peer.";
  type enumeration {
    enum "unknown";
    enum "up";
    enum "down";
  }
}

typedef actions-enum {
  description
    "Permit/deny action.";
  type enumeration {
    enum "permit";
    enum "deny";
  }
}

grouping ACTIONS {
  description
    "Permit/deny action.";
  leaf action {
    type actions-enum;
    mandatory true;
  }
}

grouping slow-peer-config {
  description
    "Configure a slow-peer.";
  container detection {
    leaf enable {
      type boolean;
      default "true";
    }
    leaf threshold {
      type uint16 {
        range "120..3600";
      }
    }
  }
  leaf split-update-group {
    type enumeration {
      enum "dynamic";
      enum "static";
    }
  }
}
```

```
    }

    grouping update-group-management {
      description
        "Manage peers in BGP update group.";
      leaf split-as-override {
        description
          "Keeps peers with as-override in different update groups.";
        type boolean;
      }
    }

    grouping neighbour-base-af-config {
      description
        "A set of configuration parameters that is applicable to all neighbour a
ddress families.";
      leaf active {
        description
          "Enable the address family for this neighbor.";
        type boolean;
        default "false";
      }
      leaf advertisement-interval {
        description
          "Minimum interval between sending BGP routing updates.";
        type uint32;
      }
      leaf allowas-in {
        description
          "Accept as-path with my AS present in it.";
        type boolean;
        default "false";
      }
      leaf maximum-prefix {
        description
          "Maximum number of prefixes accepted from this peer.";
        type uint32;
      }
      leaf next-hop-self {
        description
          "Enable the next hop calculation for this neighbor.";
        type boolean;
        default "true";
      }
      leaf next-hop-unchanged {
        description
          "Propagate next hop unchanged for iBGP paths to this neighbour.";
        type boolean;
        default "true";
      }
    }
  }
}
```

```

    }
    container remove-private-as {
      leaf remove-private-as-number {
        description
          "Remove private AS number from outbound updates.";
        type boolean;
      }
      leaf replace-with-local-as {
        description
          "Replace private AS number with local AS.";
        type boolean;
      }
    }
    leaf route-reflector-client {
      description
        "Configure a neighbor as Route Reflector client.";
      type boolean;
      default "false";
    }
    leaf send-community {
      description
        "Send Community attribute to this neighbor.";
      type enumeration {
        enum "both";
        enum "extended";
        enum "standard";
      }
      default "standard";
    }
    uses slow-peer-config;
    leaf soo {
      description
        "Site-of-Origin extended community. Format is ASN:nn or IP-address:nn"
;
      type string;
    }
    leaf weight {
      description
        "Set default weight for routes from this neighbor.";
      type uint16;
    }
  }

  grouping neighbour-common-af-config {
    description
      "A set of configuration parameters that is applicable to all neighbour a
ddress families,
      except of nsap and rtfiler.";
    uses neighbour-base-af-config;
    leaf prefix-list {

```



```
        description
          "Reference to the prefix list of this neighbour.";
        type prefix-list-ref;
      }
      leaf soft-reconfiguration {
        description
          "Allow inbound soft reconfiguration.";
        type boolean;
      }
    }

    grouping neighbour-cast-af-config {
      description
        "A set of configuration parameters that is applicable to both unicast and
        d multicast sub-address families.";
      uses neighbour-common-af-config;
      leaf propagate-dmzlink-bw {
        description
          "Propagate the DMZ link bandwidth.";
        type boolean;
      }
      container default-originate {
        description
          "Originate default route to this neighbor.";
        leaf enable {
          type boolean;
          default "false";
        }
      }
    }

    grouping neighbour-ip-multicast-af-config {
      description
        "A set of configuration parameters that is applicable to ip multicast.";
      uses neighbour-cast-af-config;
      leaf route-server-client-context {
        description
          "Specifies Route Server client context name.";
        type string;
      }
    }

    grouping neighbour-ip-unicast-af-config {
      description
        "A set of configuration parameters that is applicable to ip unicast.
        This grouping is intended to be extended by vendors as necessary to describe
        the vendor-specific configuration parameters.";
      uses neighbour-ip-multicast-af-config;
    }
  }
}
```

```
grouping bgp-af-config {
  description
    "A set of configuration parameters that is applicable to all address families of the BGP router.";
  leaf additional-paths {
    description
      "Additional paths in the BGP table.";
    type enumeration {
      enum "all";
      enum "best-n";
      enum "group-best";
    }
  }
  leaf advertise-best-external {
    description
      "Advertise best external path to internal peers.";
    type boolean;
  }
  container aggregate-timer {
    description
      "Configure aggregation timer.";
    leaf enable {
      type boolean;
      default "true";
    }
    leaf threshold {
      type uint16 {
        range "6..60";
      }
    }
  }
  container bestpath {
    description
      "Change the default bestpath selection.";
    choice bestpath-selection {
      case as-path {
        description
          "Configures a BGP router to not consider the autonomous system (AS) path during best path route selection.";
        leaf ignore-as-path {
          type boolean;
          default "false";
        }
      }
      case compare-routerid {
        description
          "Configures a BGP router to compare identical routes received from different external peers during the best path selection process and to select the route with the lowest router ID as the best path.";
        leaf ignore-routerid {
          type boolean;
        }
      }
    }
  }
}
```

```

        default "false";
    }
}
case cost-community {
    description
        "Configures a BGP router to not evaluate the cost community attrib
ute
        during the best path selection process.";
    leaf ignore-cost-community {
        type boolean;
        default "false";
    }
}
case igp-metric {
    description
        "Configures the system to ignore the IGP metric during BGP best pa
th selection.";
    leaf ignore-igp-metric {
        type boolean;
        default "false";
    }
}
case mad-confed {
    description
        "Configure a BGP routing process to compare the Multi Exit Discrim
inator (MED)
        between paths learned from confederation peers.";
    leaf enable {
        type boolean;
        default "false";
    }
    leaf missing-as-worst {
        description
            "Assigns a value of infinity to routes that are missing
            the Multi Exit Discriminator (MED) attribute,
            making the path without a MED value the least desirable path";
        type boolean;
        default "false";
    }
}
}
}
leaf dampening {
    description
        "Enable route-flap dampening.";
    type boolean;
    default "false";
}
leaf propagate-dmzlink-bw {
    description
        "Use DMZ Link Bandwidth as weight for BGP multipaths.";

```

```
        type boolean;
    }
    leaf redistribute-internal {
        description
            "Allow redistribution of iBGP into IGPs (dangerous)";
        type boolean;
    }
    leaf scan-time {
        description
            "Configure background scanner interval in seconds.";
        type uint8 {
            range "5..60";
        }
    }
    uses slow-peer-config;
    leaf soft-reconfig-backup {
        description
            "Use soft-reconfiguration inbound only when route-refresh is not negot
iated.";
        type boolean;
    }
}

grouping bgp-af-vpn-config {
    description
        "A set of configuration parameters that is applicable to vpn sub-address
family on the BGP router.";
    uses bgp-af-config;
    uses update-group-management;
}

grouping bgp-af-mvpn-config {
    description
        "A set of configuration parameters that is applicable to mvpn sub-address
s family on the BGP router.";
    leaf scan-time {
        description
            "Configure background scanner interval in seconds.";
        type uint8 {
            range "5..60";
        }
    }
    uses slow-peer-config;
    leaf soft-reconfig-backup {
        description
            "Use soft-reconfiguration inbound only when route-refresh is not negot
iated.";
        type boolean;
    }
    leaf propagate-dmzlink-bw {
        description
            "Use DMZ Link Bandwidth as weight for BGP multipaths.";
    }
}
```

```
        type boolean;
    }
    leaf rr-group {
        description
            "Extended community list name.";
        type string;
    }
    uses update-group-management;
}

grouping redistribute {
    description
        "Redistribute information from another routing protocol.
        This grouping is intended to be augmented by vendors to implement vendor-specific
        protocol redistribution configuration options.";
    choice protocol {
        case bgp {
            leaf enable-bgp {
                type boolean;
            }
        }
        case ospf {
            leaf enable-ospf {
                type boolean;
            }
        }
        case isis {
            leaf enable-isis {
                type boolean;
            }
        }
        case connected {
            leaf enable-connected {
                type boolean;
            }
        }
        case eigrp {
            leaf enable-eigrp {
                type boolean;
            }
        }
        case mobile {
            leaf enable-mobile {
                type boolean;
            }
        }
        case static {
            leaf enable-static {
                type boolean;
            }
        }
    }
}
```

```
    }
  }
  case rip {
    leaf enable-rip {
      type boolean;
    }
  }
}

grouping router-af-config {
  description
    "A set of configuration parameters that is applicable to all address families on the BGP router.";
  leaf aggregate-address {
    description
      "Configure BGP aggregate address.";
    type inet:ip-address;
  }
  leaf distance {
    description
      "Define an administrative distance.";
    type uint8 {
      range "1..255";
    }
  }
  leaf network {
    description
      "Specify a network to announce via BGP.";
    type inet:ip-address;
  }
  uses redistribute;
}

grouping maximum-paths {
  description
    "Configures packet forwarding over multiple paths.";
  leaf number-of-path {
    type uint8 {
      range "1..32";
    }
  }
  leaf ibgp-number-of-path {
    type uint8 {
      range "1..32";
    }
  }
}
```

```
container bgp-router {
  description
    "This is a top-level container for the BGP router.";
  leaf local-as-number {
    type uint32;
  }
  leaf local-as-identifier {
    type inet:ip-address;
  }
  container rpki-config {
    description
      "RPKI configuration parameters.";
    container cache-server-config {
      description
        "Configure the RPKI cache-server parameters in rpki-server configura
tion mode.";
      choice server {
        case ip-address {
          leaf ip-address {
            type inet:ip-address;
            mandatory true;
          }
        }
        case host-name {
          leaf ip-host-address {
            type inet:host;
            mandatory true;
          }
        }
      }
    }
    choice transport {
      description
        "Specifies a transport method for the RPKI cache.";
      case tcp {
        leaf tcp-port {
          type uint32;
        }
      }
      case ssh {
        leaf ssh-port {
          type uint32;
        }
      }
    }
    leaf user-name {
      type string;
    }
    leaf password {
      type string;
    }
  }
}
```

```
    }
    leaf preference-value {
      description
        "Specifies a preference value for the RPKI cache.
        Setting a lower preference value is better.";
      type uint8 {
        range "1..10";
      }
    }
    leaf purge-time {
      description
        "Configures the time BGP waits to keep routes from a cache after the
        cache session drops. Set purge time in seconds.";
      type uint16 {
        range "30..360";
      }
    }
    choice refresh-time {
      description
        "Configures the time BGP waits in between sending periodic serial
        queries to the cache. Set refresh-time in seconds.";
      case disable {
        leaf refresh-time-disable {
          type boolean;
        }
      }
      case set-time {
        leaf refresh-interval {
          type uint16 {
            range "15..3600";
          }
        }
      }
    }
    choice response-time {
      description
        "Configures the time BGP waits for a response after sending a serial
        or reset query. Set response-time in seconds.";
      case disable {
        leaf response-time-disable {
          type boolean;
        }
      }
      case set-time {
        leaf response-interval {
          type uint16 {
            range "15..3600";
          }
        }
      }
    }
  }
}
```



```
    }
    container validation-config {
      description
        "Controls the behavior of RPKI prefix validation processing.";
      leaf enable {
        description
          "Enables RPKI origin-AS validation.";
        type boolean;
        default "true";
      }
      leaf enable-ibgp {
        description
          "Enables the iBGP signaling of validity state through an extended-
community.";
        type boolean;
      }
      choice validation-time {
        description
          "Sets prefix validation time (in seconds) or to set off the automa
tic prefix validation after an RPKI update.";
        case validation-off {
          leaf disable {
            type boolean;
          }
        }
        case set-time {
          leaf prefix-validation-time {
            description
              "Range in seconds.";
            type uint16 {
              range "5..60";
            }
          }
        }
      }
    }
  }
  container bestpath-computation {
    description
      "Configures RPKI bestpath computation options.";
    leaf enable {
      description
        "Enables the validity states of BGP paths to affect the path's pre
ference in the BGP bestpath process.";
      type boolean;
    }
    leaf allow-invalid {
      description
        "Allows all 'invalid' paths to be considered for BGP bestpath comp
utation.";
      type boolean;
    }
  }
}
```

```
    }
    container af-configuration {
      description
        "Top level container for address families specific configuration of the BGP router.";
      container ipv4 {
        container mdt {
          container bgp {
            description
              "BGP specific commands for ipv4-mdt address family/sub-address family combination.";
            leaf dampening {
              description
                "Enable route-flap dampening.";
              type boolean;
              default "false";
            }
            leaf scan-time {
              description
                "Configure background scanner interval in seconds.";
              type uint8 {
                range "5..60";
              }
            }
          }
          uses slow-peer-config;
          leaf soft-reconfig-backup {
            description
              "Use soft-reconfiguration inbound only when route-refresh is not negotiated.";
            type boolean;
          }
          leaf propagate-dmzlink-bw {
            description
              "Use DMZ Link Bandwidth as weight for BGP multipaths.";
            type boolean;
          }
        }
      }
      container multicast {
        container bgp {
          description
            "BGP specific commands for ipv4-multicast address family/sub-address family combination.";
          uses bgp-af-config;
        }
        leaf auto-summary {
          description
            "Enable automatic network number summarization";
          type boolean;
        }
        uses router-af-config;
        leaf default-metric {
```

```

        description
            "Set metric of redistributed routes.";
        type uint32;
    }
}
container unicast {
    container bgp {
        description
            "BGP specific commands for ipv4-unicast address family/sub-address family combination.";
        uses bgp-af-config;
        leaf always-compare-med {
            description
                "Allow comparing MED from different neighbors.";
            type boolean;
            default "false";
        }
        leaf enforce-first-as {
            description
                "Enforce the first AS for EBGp routes(default).";
            type boolean;
            default "true";
        }
        leaf fast-external-fallover {
            description
                "Immediately reset session if a link to a directly connected external peer goes down.";
            type boolean;
            default "true";
        }
        leaf suppress-inactive {
            description
                "Suppress routes that are not in the routing table.";
            type boolean;
        }
        leaf asnotation {
            description
                "Sets the default asplain notation.";
            type enumeration {
                enum "asplain";
                enum "dot";
            }
        }
        leaf enable-client-to-client-reflection {
            description
                "Manages client to client route reflection.";
            type boolean;
            default "true";
        }
        leaf cluster-id {

```

```
    description
      "Configure Route-Reflector Cluster-id.";
    type string;
  }
  container confederation {
    description
      "AS confederation parameters.";
    leaf identifier {
      description
        "Confederation identifier.";
      type string;
    }
    list peers {
      description
        "Confederation peers.";
      key "as-name";
      leaf as-name {
        type string;
      }
    }
  }
  container consistency-checker {
    description
      "Consistency-checker configuration.";
    leaf enable {
      type boolean;
    }
    leaf interval {
      description
        "Check interval in minutes.";
      type uint16 {
        range "5..1440";
      }
    }
  }
  choice inconsistency-action {
    case error-message {
      description
        "Specifies that when an inconsistency is found, the system
will only generate a syslog message.";
      leaf generate-error-message-only {
        type boolean;
      }
    }
    case autorepair {
      description
        "Specifies that when an inconsistency is found,
the system will generate a syslog message and take action
based on the type of inconsistency found.";
      leaf perform-autorepair {
```

```

        type boolean;
    }
}
}
leaf deterministic-med {
    description
        "If enabled it enforce the deterministic comparison of the MED
value between
        all paths received from within the same autonomous system.";
    type boolean;
}
container graceful-restart {
    description
        "Controls the BGP graceful restart capability.";
    leaf enable {
        type boolean;
    }
    leaf restart-time {
        description
            "Sets the maximum time period (in seconds) that the local ro
uter will wait
            for a graceful-restart-capable neighbor to return to normal
operation after a restart event occurs.";
        type uint16 {
            range "1..3600";
        }
        default "120";
    }
    leaf stalepath-time {
        description
            "Sets the maximum time period that the local router will hol
d stale paths for a restarting peer.";
        type uint16 {
            range "5..3600";
        }
        default "360";
    }
}
container listener-congfig {
    description
        "Associates a subnet range with a BGP peer group and activate
the BGP dynamic neighbors feature.";
    leaf enable {
        type boolean;
    }
    leaf limit {
        description
            "Sets a maximum limit number of BGP dynamic subnet range nei
ghbors.";
        type uint16 {
            range "1..5000";
        }
        default "100";
    }
}

```



```

    }
    leaf range {
      description
        "Specifies a subnet range that is to be associated with a sp
ecified peer group.";
      type uint16 {
        range "0..32";
      }
    }
    leaf peer-group {
      description
        "Specifies a BGP peer group that is to be associated with th
e specified subnet range.";
      type string;
    }
  }
  leaf log-neighbor-changes {
    description
      "Log neighbor up/down and reset reason.";
    type boolean;
  }
  leaf max-as-limit {
    description
      "Configures BGP to discard routes that have a number of autono
mous system numbers in AS-path that exceed the specified value.";
    type uint16 {
      range "1..254";
    }
  }
  container router-id {
    description
      "Configures a fixed router ID for the local BGP routing proces
s.";
    leaf enable {
      type boolean;
    }
    choice config-type {
      case static {
        leaf ip-address {
          type boolean;
        }
      }
      case auto-config {
        leaf enable-auto-config {
          type boolean;
        }
      }
    }
  }
  container transport {
    description
      "Manages transport session parameters.";

```

```

        leaf enable-path-mtu-discovery {
            description
                "Enables transport path MTU discovery.";
            type boolean;
            default "true";
        }
    }
    leaf auto-summary {
        description
            "Enable automatic network number summarization";
        type boolean;
    }
    uses router-af-config;
    uses maximum-paths;
    leaf synchronization {
        description
            "Perform IGP synchronization.";
        type boolean;
    }
}
container mvpn {
    container bgp {
        description
            "BGP specific commands for ipv4-mvpn address family/sub-address f
amily combination.";
        uses bgp-af-mvpn-config;
    }
    leaf auto-summary {
        description
            "Enable automatic network number summarization.";
        type boolean;
    }
}
}
container ipv6 {
    container multicast {
        container bgp {
            description
                "BGP specific commands for ipv6-multicast address family/sub-add
ess family combination.";
            uses bgp-af-config;
        }
        uses router-af-config;
    }
    container unicast {
        container bgp {
            description
                "BGP specific commands for ipv6-unicast address family/sub-addes
s family combination.";
            uses bgp-af-config;
        }
    }
}

```



```
    }
    uses router-af-config;
    leaf default-metric {
        description
            "Set metric of redistributed routes.";
        type uint32;
    }
    uses maximum-paths;
    leaf synchronization {
        description
            "Perform IGP synchronization.";
        type boolean;
    }
}
container mvpn {
    container bgp {
        description
            "BGP specific commands for ipv6-mvpn address family/sub-address f
amily combination.";
        uses bgp-af-mvpn-config;
    }
}
container l2vpn {
    container vpls {
        container bgp {
            description
                "BGP specific commands for l2vpn-vpls address family/sub-address
family combination.";
            leaf scan-time {
                description
                    "Configure background scanner interval in seconds.";
                type uint8 {
                    range "5..60";
                }
            }
            uses slow-peer-config;
        }
    }
}
container nsap {
    container unicast {
        container bgp {
            description
                "BGP specific commands for nsap-unicast address family/sub-addres
s family combination.";
            container aggregate-timer {
                description
                    "Configure Aggregation Timer.";
            }
            leaf enable {
                type boolean;
            }
        }
    }
}
```

```
        default "true";
    }
    leaf threshold {
        type uint16 {
            range "6..60";
        }
    }
}
leaf dampening {
    description
        "Enable route-flap dampening.";
    type boolean;
    default "false";
}
leaf propagate-dmzlink-bw {
    description
        "Use DMZ Link Bandwidth as weight for BGP multipaths.";
    type boolean;
}
leaf redistribute-internal {
    description
        "Allow redistribution of iBGP into IGPs (dangerous)";
    type boolean;
}
leaf scan-time {
    description
        "Configure background scanner interval in seconds.";
    type uint8 {
        range "5..60";
    }
}
}
uses slow-peer-config;
leaf soft-reconfig-backup {
    description
        "Use soft-reconfiguration inbound only when route-refresh is not negotiated.";
    type boolean;
}
}
leaf default-metric {
    description
        "Set metric of redistributed routes.";
    type uint32;
}
uses maximum-paths;
leaf network {
    description
        "Specify a network to announce via BGP.";
    type inet:ip-address;
```

```
    }
    uses redistribute;
    leaf synchronization {
        description
            "Perform IGP synchronization.";
        type boolean;
    }
}
}
}
container rtfiler {
    container unicast {
        container bgp {
            description
                "BGP specific commands for rtfiler-unicast address family/sub-add
ess family combination.";
            uses slow-peer-config;
        }
        uses maximum-paths;
    }
}
container vpnv4 {
    container unicast {
        container bgp {
            description
                "BGP specific commands for vpnv4-unicast address family/sub-address
family combination.";
            uses bgp-af-vpn-config;
        }
        uses maximum-paths;
    }
    container multicast {
        container bgp {
            description
                "BGP specific commands for vpnv4-multicast address family/sub-adde
ss family combination.";
            uses bgp-af-vpn-config;
        }
        uses maximum-paths;
    }
}
container vpnv6 {
    container unicast {
        container bgp {
            description
                "BGP specific commands for vpnv6-unicast address family/sub-address
family combination.";
            uses bgp-af-vpn-config;
        }
    }
}
}
```

```
container bgp-neighbors {
  description
    "The top level container for the list of neighbours of the BGP router.";
  list bgp-neighbor {
    key "as-number";
    leaf as-number {
      type uint32;
    }
    choice peer-address-type {
      case ip-address {
        leaf ip-address {
          type inet:ip-address;
          mandatory true;
        }
      }
      case prefix {
        leaf prefix {
          type inet:ip-prefix;
          mandatory true;
        }
      }
      case host {
        leaf ip-host-address {
          type inet:host;
          mandatory true;
        }
      }
    }
  }
  leaf prefix-list {
    type prefix-list-ref;
  }
  leaf default-action {
    type actions-enum;
  }
  container af-specific-config {
    description
      "Address family specific configuration parameters for the neighbours
.";
    container ipv4 {
      container mdt {
        uses neighbour-common-af-config;
      }
      container unicast {
        uses neighbour-ip-unicast-af-config;
      }
      container multicast {
        uses neighbour-ip-multicast-af-config;
      }
      container mvpn {
```

```
        uses neighbour-cast-af-config;
    }
}
container ipv6 {
    container unicast {
        uses neighbour-ip-unicast-af-config;
    }
    container multicast {
        uses neighbour-ip-multicast-af-config;
    }
    container mvpn {
        uses neighbour-common-af-config;
    }
}
container l2vpn {
    container evpn {
        uses neighbour-common-af-config;
    }
    container vpls {
        uses neighbour-common-af-config;
    }
}
container nsap {
    container unicast {
        uses neighbour-base-af-config;
        leaf prefix-list {
            type prefix-list-ref;
        }
    }
}
container rtfiler {
    container unicast {
        uses neighbour-base-af-config;
        leaf soft-reconfiguration {
            description
                "Allow inbound soft reconfiguration.";
            type boolean;
        }
    }
}
container vpnv4 {
    container unicast {
        uses neighbour-cast-af-config;
    }
    container multicast {
        uses neighbour-cast-af-config;
    }
}
```

```

        container vpnv6 {
            container unicast {
                uses neighbour-cast-af-config;
            }
            container multicast {
                uses neighbour-cast-af-config;
            }
        }
    }
    container bgp-neighbor-state {
        description
            "The operational parameters describing the neighbour state.
            It is intended that this container may be augmented by vendors to r
eflect the vendor-specific operational state parameters.";
        leaf adminStatus {
            type bgp-peer-admin-status;
        }
        leaf in-lastupdatetime {
            type yang:timestamp;
        }
    }
    container bgp-neighbor-statistics {
        description
            "The operational parameters describing the neighbour statistics.
            It is intended that this container may be augmented by vendors to r
eflect the vendor-specific staistical parameters.";
        leaf nr-in-updates {
            type uint32;
        }
        leaf nr-out-updates {
            type uint32;
        }
    }
}
}
}
container prefix-lists {
    description
        "Contains all prefix lists defined
        on a router.";
    list prefix-list {
        key "prefix-list-name";
        description
            "A prefix list.";
        leaf prefix-list-name {
            type string;
        }
        container prefixes {
            list prefix {
                key "seq-nr";
                description

```

```
"A prefix is a rule with a BGP filter.
The left hand side of the rule is the prefix filter.
It specifies a set of IP addresses.
If a BGP announcement contains an address that matches, the
rule is applied. The right hand side of the rule specifies
the action that is to be applied.";
leaf seq-nr {
  type uint16;
  description
    "Sequence number of the rule.
    The sequence number is included for compatibility purposes
    with CLI; from a machine-to-machine interface perspective,
    it would strictly speaking not be required as list elements
    can be arranged in a particular order.";
}
container prefix-filter {
  choice ip-address-group {
    case ip-address {
      leaf ip-address {
        type inet:ip-address;
        mandatory true;
      }
    }
    case prefix {
      leaf prefix {
        type inet:ip-prefix;
        mandatory true;
      }
    }
    case host {
      leaf ip-host-address {
        type inet:host;
        mandatory true;
      }
    }
    case ip-range {
      leaf lower {
        type inet:ip-address;
      }
      leaf upper {
        type inet:ip-address;
      }
    }
  }
}
leaf action {
  type actions-enum;
  mandatory true;
  description
```

[illegible]

5. IANA Considerations

6. Security Considerations

The transport protocol used for sending the BGP data **MUST** support authentication and **SHOULD** support encryption. The data-model by itself does not create any security implications.

This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

7. Acknowledgements

The authors would like to thank the reviewers of this document for their comments.

8. References

8.1. Normative References

```
[I-D.ietf-netmod-routing-cfg]
    Lhotka, L., "A YANG Data Model for Routing Management",
    draft-ietf-netmod-routing-cfg-09 (work in progress),
    February 2013.
```

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

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

- [RFC2842] Chandra, R. and J. Scudder, "Capabilities Advertisement with BGP-4", RFC 2842, May 2000.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

8.2. Informative References

- [I-D.ietf-netmod-interfaces-cfg]
Bjorklund, M., "A YANG Data Model for Interface Management", draft-ietf-netmod-interfaces-cfg-09 (work in progress), February 2013.
- [RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, February 2009.

Authors' Addresses

Aleksandr Zhdankin
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: azhdanki@cisco.com

Keyur Patel
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: keyupate@cisco.com

Alexander Clemm
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: alex@cisco.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2014

C. Zhou
Huawei Technologies
Q. Sun
China Telecom
October 21, 2013

A YANG Data Model for Open IPv6 Transition
draft-zhou-netmod-openv6-transition-cfg-00

Abstract

During the transition from IPv4 to IPv6, there are many kinds of transition methods and scenarios that have been or are currently being defined in IETF, e.g., DS-Lite, Lw4over6, MAP-E, 4rd, 6rd and etc. Carriers have to select and determine their transition ways among all of these techniques, which brings slow transition to IPv6. Currently, we face two main challenges in IPv6 transition: the legacy equipment does not support multiple IPv6 transition technologies at the same time and there are not enough native IPv6 applications. This document describes an open IPv6 YANG [RFC6020] data model which serves as a framework for configuring and managing an IPv6 service to provide a low-cost and unified way to IPv6 transition.

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, 2014.

Copyright Notice

Copyright (c) 2013 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	2
1.1. Requirements Language	2
2. Terminology	2
3. Objectives	3
4. The Design of the Open IPv6 Transition Data Model	3
4.1. Flow Table	4
4.2. Resource	5
5. Acknowledgements	5
6. IANA Considerations	5
7. Security Considerations	5
8. Normative References	5
Authors' Addresses	5

1. Introduction

Each IPv6 transition technology has its own specific characteristics. In order to support a specific IPv6 transition scheme and not need to upgrade the devices, we propose two elements for OSS to configure the IPv6 service related parameters: the resource template and the flow table template. These two elements together define the so-called open IPv6 data model, which is proposed as a basis for the development of data models for configuration and management of the specific IPv6 transition technology. This data model could accommodate existing and future IPv6 transition schemes.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. Terminology

The following terms are used in this document:

- o Resource template: the template for IP pool and port/port set;

- o Flow table template: the template for various transition technologies, described by NAT and/or tunnel.

3. Objectives

The initial design of the open IPv6 data model was driven by the following objectives:

- o The data model should support the particular technology (any of the v4-v6 transition technology). The resource template and the flow table template should be general templates suitable for all the transition schemes;
- o The users/applications should be able to decide for themselves when and how to start the IPv6 transition;
- o The data model is generic and protocol-independent for packets and flow-tables interaction between network devices and applications.
- o Provides the management functions (e.g.,OSS) for the applications to configure and manage the installed modules in the network devices.

4. The Design of the Open IPv6 Transition Data Model

The open IPv6 transition data model consists of two YANG modules. The first module, "flow table", defines the action (tunnel and NAT) of a specific transition technology. The other module, "resource", defines the general IP prefix and port set/mapping rules for a specific transition technology. Figure 1 show abridged views of the configuration and operational state data hierarchies.

```

+--rw ipv6-transition
  +--rw flow table [name]
    |   +--rw name                string
    |   +--rw description?        string
    |   +--rw enabled?            boolean
    |   +--rw tunnel
    |     |   +--rw name                string
    |     |   +--rw description?        string
    |     |   +--rw enabled?            boolean
    |     |   +--rw type                identityref
    |     |   +--rw inner-ip-address-src inet:ip-address
    |     |   +--rw inner-ip-address-dst inet:ip-address
    |     |   +--rw outer-ip-address-src inet:ip-address
    |     |   +--rw outer-ip-address-dst inet:ip-address
    |   +--rw translation
    |     +--rw name                string

```

```

|      +--rw description?          string
|      +--rw enabled?             boolean
|      +--rw type                  identityref
|      +--rw proto?               string
|      +--rw inner-ip-address      inet:ip-address
|      +--rw outer-ip-address      inet:ip-address
|      +--rw inner-port?           inet:port-number
|      +--rw outer-port?           inet:port-number
+--rw resource [name]
|   +--rw name
|   +--rw description?            string
|   +--rw enabled?                boolean
|   +--rw ip prefix                inet:ip-prefix
|   |
|   +--rw port set                 inet:port-set

```

Figure 1: Open IPv6 Configuration data hierarchy.

4.1. Flow Table

The flow table element defines the two basic components of a specific IPv6 transition technology: tunnel and translation.

The main attributes of the tunnel are explained as below:

- o "inner-ip-address-src": the source ip address of a packet or inner source ip address when the packet type is IP-in-IP tunnel;
- o "inner-ip-address-dst": the destination ip address of a packet or inner destination ip address when the packet type is IP-in-IP tunnel;
- o "outer-ip-address-src": the source ip address of a packet or outer source ip address when the packet type is IP-in-IP tunnel;
- o "outer-ip-address-dst": the destination ip address of a packet or outer destination ip address when the packet type is IP-in-IP tunnel.

The IP address could be IPv4 or IPv6 depending on the technologies.

The main attributes of the translation are explained as below:

- o "inner-ip-address": the source ip address of a packet;
- o "outer-ip-addresss": the destination ip address of a packet;

- o "inner-port": the source port of a packet;
- o "outer-port": the destination port of a packet.

The IP address could be IPv4 or IPv6 depending on the specific technologies.

4.2. Resource

The resource element defines the general IP pool and the port set of the specific transition technology if the ipv4 address sharing mechanism is adopted.

5. Acknowledgements

TBD

6. IANA Considerations

7. Security Considerations

To come.

8. 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.
- [RFC6431] Boucadair, M., Levis, P., Bajko, G., Savolainen, T., and T. Tsou, "Huawei Port Range Configuration Options for PPP IP Control Protocol (IPCP)", RFC 6431, November 2011.

Authors' Addresses

Cathy Zhou
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: cathy.zhou@huawei.com

Qiong Sun
China Telecom
P.R.China

Phone: 86 10 58552936
Email: sunqiong@ctbri.com.cn