Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

A. Bierman
YumaWorks

February 14, 2014

Guidelines for Authors and Reviewers of YANG Data Model Documents
draft-bierman-netmod-rfc6087bis-00

Abstract

This memo provides guidelines for authors and reviewers of Standards Track specifications containing YANG data model modules. Applicable portions may be used as a basis for reviews of other YANG data model documents. Recommendations and procedures are defined, which are intended to increase interoperability and usability of Network Configuration Protocol (NETCONF) implementations that utilize 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 August 18, 2014.

Copyright Notice

Table of Contents

1. Introduction

The standardization of network configuration interfaces for use with the Network Configuration Protocol [RFC6241] requires a modular set of data models, which can be reused and extended over time.

This document defines a set of usage guidelines for Standards Track documents containing [RFC6020] data models. YANG is used to define the data structures, protocol operations, and notification content used within a NETCONF server. A server that supports a particular YANG module will support client NETCONF operation requests, as indicated by the specific content defined in the YANG module.

This document is similar to the Structure of Management Information version 2 (SMIv2) usage guidelines specification [RFC4181] in intent and structure. However, since that document was written a decade after SMIv2 modules had been in use, it was published as a 'Best Current Practice' (BCP). This document is not a BCP, but rather an informational reference, intended to promote consistency in documents containing YANG modules.

Many YANG constructs are defined as optional to use, such as the description statement. However, in order to maximize interoperability of NETCONF implementations utilizing YANG data models, it is desirable to define a set of usage guidelines that may require a higher level of compliance than the minimum level defined in the YANG specification.

In addition, YANG allows constructs such as infinite length identifiers and string values, or top-level mandatory nodes, that a compliant server is not required to support. Only constructs that all servers are required to support can be used in IETF YANG modules.

This document defines usage guidelines related to the NETCONF operations layer and NETCONF content layer, as defined in [RFC6241]. These guidelines are intended to be used by authors and reviewers to improve the readability and interoperability of published YANG data models.

2. Terminology

2.1. Requirements Notation

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

RFC 2119 language is used here to express the views of the NETMOD working group regarding content for YANG modules. YANG modules complying with this document will treat the RFC 2119 terminology as if it were describing best current practices.

2.2. NETCONF Terms

The following terms are defined in [RFC6241] and are not redefined here:

- capabilities
- client
- operation
- server

2.3. YANG Terms

The following terms are defined in [RFC6020] and are not redefined here:

- data node
- module
- namespace
- submodule
- version
- YANG
- YIN

Note that the term ’module’ may be used as a generic term for a YANG module or submodule. When describing properties that are specific to submodules, the term ’submodule’ is used instead.

2.4. Terms

The following terms are used throughout this document:

- published: A stable release of a module or submodule, usually contained in an RFC.
- unpublished: An unstable release of a module or submodule, usually contained in an Internet-Draft.

3. General Documentation Guidelines

YANG data model modules under review are likely to be contained in Internet-Drafts. All guidelines for Internet-Draft authors MUST be followed. The RFC Editor provides guidelines for authors of RFCs, which are first published as Internet-Drafts. These guidelines should be followed and are defined in [RFC2223] and updated in [RFC5741] and "RFC Document Style" [RFC-STYLE].

The following sections MUST be present in an Internet-Draft containing a module:

- Narrative sections
- Definitions section
- Security Considerations section
- IANA Considerations section
- References section

3.1. Module Copyright

The module description statement MUST contain a reference to the latest approved IETF Trust Copyright statement, which is available online at:

http://trustee.ietf.org/license-info/

Each YANG module or submodule contained within an Internet-Draft or RFC is considered to be a code component. The strings '<CODE BEGINS>' and '<CODE ENDS>' MUST be used to identify each code component.

The '<CODE BEGINS>' tag SHOULD be followed by a string identifying the file name specified in Section 5.2 of [RFC6020]. The following example is for the '2010-01-18' revision of the 'ietf-foo' module:

```
  <CODE BEGINS> file "ietf-foo@2010-01-18.yang"
    module ietf-foo {
      // ...
      revision 2010-01-18 {
        description "Latest revision";
        reference "RFC XXXX";
      }
      // ...
    }
```

<CODE ENDS>

3.2. Terminology Section

A terminology section MUST be present if any terms are defined in the document or if any terms are imported from other documents.

If YANG tree diagrams are used, then a sub-section explaining the YANG tree diagram syntax MUST be present, containing the following text:

> A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:
>
> - Brackets "[" and "]" enclose list keys.
> - Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
> - Symbols after data node names: "?" means an optional node and "*" denotes a "list" and "leaf-list".
> - Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
> - Ellipsis ("...") stands for contents of subtrees that are not shown.

3.3. Tree Diagrams

YANG tree diagrams provide a concise representation of a YANG module, and SHOULD be included to help readers understand YANG module structure. Tree diagrams MAY be split into sections to correspond to document structure.

The following example shows a simple YANG tree diagram:

```
   +--rw top-level-config-container
   |  +--rw config-list* [key-name]
   |     +--rw key-name                    string
   |     +--rw optional-parm?              string
   |     +--rw mandatory-parm              identityref
   |     +--ro read-only-leaf              string
   +--ro top-level-nonconfig-container
      +--ro nonconfig-list* [name]
         +--ro name               string
         +--ro type               string
```

3.4. Narrative Sections

The narrative part MUST include an overview section that describes the scope and field of application of the module(s) defined by the specification and that specifies the relationship (if any) of these modules to other standards, particularly to standards containing other YANG modules. The narrative part SHOULD include one or more sections to briefly describe the structure of the modules defined in the specification.

If the module(s) defined by the specification imports definitions from other modules (except for those defined in the [RFC6020] or [RFC6991] documents), or are always implemented in conjunction with other modules, then those facts MUST be noted in the overview section, as MUST be noted any special interpretations of definitions in other modules.

3.5. Definitions Section

This section contains the module(s) defined by the specification. These modules MUST be written using the YANG syntax defined in [RFC6020]. A YIN syntax version of the module MAY also be present in the document. There MAY also be other types of modules present in the document, such as SMIv2, which are not affected by these guidelines.

See Section 4 for guidelines on YANG usage.

3.6. Security Considerations Section

Each specification that defines one or more modules MUST contain a section that discusses security considerations relevant to those modules.

This section MUST be patterned after the latest approved template (available at http://trac.tools.ietf.org/area/ops/trac/wiki/yang-security-guidelines). Section 6.1 contains the security considerations template dated 2010-06-16. Authors MUST check the webpage at the URL listed above in case there is a more recent version available.

In particular:

- Writable data nodes that could be especially disruptive if abused MUST be explicitly listed by name and the associated security risks MUST be explained.

- Readable data nodes that contain especially sensitive information or that raise significant privacy concerns MUST be explicitly listed by name and the reasons for the sensitivity/privacy concerns MUST be explained.

- Operations (i.e., YANG 'rpc' statements) that are potentially harmful to system behavior or that raise significant privacy concerns MUST be explicitly listed by name and the reasons for the sensitivity/privacy concerns MUST be explained.

3.7. IANA Considerations Section

In order to comply with IESG policy as set forth in http://www.ietf.org/id-info/checklist.html, every Internet-Draft that is submitted to the IESG for publication MUST contain an IANA Considerations section. The requirements for this section vary depending on what actions are required of the IANA. If there are no IANA considerations applicable to the document, then the IANA Considerations section stating that there are no actions is removed by the RFC Editor before publication. Refer to the guidelines in [RFC5226] for more details.

3.7.1. Documents that Create a New Namespace

If an Internet-Draft defines a new namespace that is to be administered by the IANA, then the document MUST include an IANA Considerations section that specifies how the namespace is to be administered.

Specifically, if any YANG module namespace statement value contained in the document is not already registered with IANA, then a new YANG Namespace registry entry MUST be requested from the IANA. The [RFC6020] specification includes the procedure for this purpose in its IANA Considerations section.

3.7.2. Documents that Extend an Existing Namespace

It is possible to extend an existing namespace using a YANG submodule that belongs to an existing module already administered by IANA. In this case, the document containing the main module MUST be updated to use the latest revision of the submodule.

3.8. Reference Sections

For every import or include statement that appears in a module contained in the specification, which identifies a module in a separate document, a corresponding normative reference to that document MUST appear in the Normative References section. The

reference MUST correspond to the specific module version actually used within the specification.

For every normative reference statement that appears in a module contained in the specification, which identifies a separate document, a corresponding normative reference to that document SHOULD appear in the Normative References section. The reference SHOULD correspond to the specific document version actually used within the specification. If the reference statement identifies an informative reference, which identifies a separate document, a corresponding informative reference to that document MAY appear in the Informative References section.

4. YANG Usage Guidelines

In general, modules in IETF Standards Track specifications MUST comply with all syntactic and semantic requirements of YANG [RFC6020]. The guidelines in this section are intended to supplement the YANG specification, which is intended to define a minimum set of conformance requirements.

In order to promote interoperability and establish a set of practices based on previous experience, the following sections establish usage guidelines for specific YANG constructs.

Only guidelines that clarify or restrict the minimum conformance requirements are included here.

4.1. Module Naming Conventions

Modules contained in Standards Track documents SHOULD be named according to the guidelines in the IANA Considerations section of [RFC6020].

A distinctive word or acronym (e.g., protocol name or working group acronym) SHOULD be used in the module name. If new definitions are being defined to extend one or more existing modules, then the same word or acronym should be reused, instead of creating a new one.

All published module names MUST be unique. For a YANG module published in an RFC, this uniqueness is guaranteed by IANA. For unpublished modules, the authors need to check that no other work in progress is using the same module name.

Once a module name is published, it MUST NOT be reused, even if the RFC containing the module is reclassified to 'Historic' status.

4.2. Identifiers

Identifiers for all YANG identifiers in published modules MUST be between 1 and 64 characters in length. These include any construct specified as an 'identifier-arg-str' token in the ABNF in Section 12 of [RFC6020].

4.3. Defaults

In general, it is suggested that substatements containing very common default values SHOULD NOT be present. The following substatements are commonly used with the default value, which would make the module difficult to read if used everywhere they are allowed.

Statement	Default Value
config	true
mandatory	false
max-elements	unbounded
min-elements	0
ordered-by	system
status	current
yin-element	false

Statement Defaults

4.4. Conditional Statements

A module may be conceptually partitioned in several ways, using the 'if-feature' and/or 'when' statements.

Data model designers need to carefully consider all modularity aspects, including the use of YANG conditional statements.

If a data definition is optional, depending on server support for a NETCONF protocol capability, then a YANG 'feature' statement SHOULD be defined to indicate that the NETCONF capability is supported within the data model.

If any notification data, or any data definition, for a non-configuration data node is not mandatory, then the server may or may not be required to return an instance of this data node. If any conditional requirements exist for returning the data node in a notification payload or retrieval request, they MUST be documented somewhere. For example, a 'when' or 'if-feature' statement could apply to the data node, or the conditional requirements could be explained in a 'description' statement within the data node or one of its ancestors (if any).

4.5. XPath Usage

This section describes guidelines for using the XML Path Language [W3C.REC-xpath-19991116] (XPath) within YANG modules.

The 'attribute' and 'namespace' axes are not supported in YANG, and MAY be empty in a NETCONF server implementation.

The 'position' and 'last' functions SHOULD NOT be used. This applies to implicit use of the 'position' function as well (e.g., '//chapter[42]'). A server is only required to maintain the relative

XML document order of all instances of a particular user-ordered list or leaf-list. The 'position' and 'last' functions MAY be used if they are evaluated in a context where the context node is a user-ordered 'list' or 'leaf-list'.

The 'preceding', and 'following' axes SHOULD NOT be used. These constructs rely on XML document order within a NETCONF server configuration database, which may not be supported consistently or produce reliable results across implementations. Predicate expressions based on static node properties (e.g., element name or value, 'ancestor' or 'descendant' axes) SHOULD be used instead. The 'preceding' and 'following' axes MAY be used if document order is not relevant to the outcome of the expression (e.g., check for global uniqueness of a parameter value).

The 'preceding-sibling' and 'following-sibling' axes SHOULD NOT used.

A server is only required to maintain the relative XML document order of all instances of a particular user-ordered list or leaf-list. The 'preceding-sibling' and 'following-sibling' axes MAY be used if they are evaluated in a context where the context node is a user-ordered 'list' or 'leaf-list'.

Data nodes that use the 'int64' and 'uint64' built-in type SHOULD NOT be used within numeric expressions. There are boundary conditions in which the translation from the YANG 64-bit type to an XPath number can cause incorrect results. Specifically, an XPath 'double' precision floating point number cannot represent very large positive or negative 64-bit numbers because it only provides a total precision of 53 bits. The 'int64' and 'uint64' data types MAY be used in numeric expressions if the value can be represented with no more than 53 bits of precision.

Data modelers need to be careful not to confuse the YANG value space and the XPath value space. The data types are not the same in both, and conversion between YANG and XPath data types SHOULD be considered carefully.

Explicit XPath data type conversions MAY be used (e.g., 'string', 'boolean', or 'number' functions), instead of implicit XPath data type conversions.

XPath expressions that contain a literal value representing a YANG identity SHOULD always include the declared prefix of the module where the identity is defined.

XPath expressions for 'when' statements MUST NOT reference the context node or any descendant nodes of the context node.

4.6. Lifecycle Management

The status statement MUST be present if its value is 'deprecated' or 'obsolete'.

The module or submodule name MUST NOT be changed, once the document containing the module or submodule is published.

The module namespace URI value MUST NOT be changed, once the document containing the module is published.

The revision-date substatement within the imports statement SHOULD be present if any groupings are used from the external module.

The revision-date substatement within the include statement SHOULD be present if any groupings are used from the external submodule.

If submodules are used, then the document containing the main module MUST be updated so that the main module revision date is equal or more recent than the revision date of any submodule that is (directly or indirectly) included by the main module.

4.7. Module Header, Meta, and Revision Statements

For published modules, the namespace MUST be a globally unique URI, as defined in [RFC3986]. This value is usually assigned by the IANA.

The organization statement MUST be present. If the module is contained in a document intended for Standards Track status, then the organization SHOULD be the IETF working group chartered to write the document.

The contact statement MUST be present. If the module is contained in a document intended for Standards Track status, then the working group web and mailing information MUST be present, and the main document author or editor contact information SHOULD be present. If additional authors or editors exist, their contact information MAY be present. In addition, the Area Director and other contact information MAY be present.

The description statement MUST be present. The appropriate IETF Trust Copyright text MUST be present, as described in Section 3.1.

If the module relies on information contained in other documents, which are not the same documents implied by the import statements present in the module, then these documents MUST be identified in the reference statement.

A revision statement MUST be present for each published version of the module. The revision statement MUST have a reference substatement. It MUST identify the published document that contains the module. Modules are often extracted from their original documents, and it is useful for developers and operators to know how to find the original source document in a consistent manner. The revision statement MAY have a description substatement.

Each new revision MUST include a revision date that is higher than any other revision date in the module. The revision date does not need to be updated if the module contents do not change in the new document revision.

It is acceptable to reuse the same revision statement within unpublished versions (i.e., Internet-Drafts), but the revision date MUST be updated to a higher value each time the Internet-Draft is re-posted.

4.8. Namespace Assignments

It is RECOMMENDED that only valid YANG modules be included in documents, whether or not they are published yet. This allows:

- the module to compile correctly instead of generating disruptive fatal errors.
- early implementors to use the modules without picking a random value for the XML namespace.
- early interoperability testing since independent implementations will use the same XML namespace value.

Until a URI is assigned by the IANA, a proposed namespace URI MUST be provided for the namespace statement in a YANG module. A value SHOULD be selected that is not likely to collide with other YANG namespaces. Standard module names, prefixes, and URI strings already listed in the YANG Module Registry MUST NOT be used.

A standard namespace statement value SHOULD have the following form:

```
<URN prefix string>:<module-name>
```

The following URN prefix string SHOULD be used for published and unpublished YANG modules:

```
urn:ietf:params:xml:ns:yang:
```

The following example URNs would be valid temporary namespace

statement values for Standards Track modules:

urn:ietf:params:xml:ns:yang:ietf-netconf-partial-lock

urn:ietf:params:xml:ns:yang:ietf-netconf-state

urn:ietf:params:xml:ns:yang:ietf-netconf

Note that a different URN prefix string SHOULD be used for non-Standards-Track modules. The string SHOULD be selected according to the guidelines in [RFC6020].

The following examples of non-Standards-Track modules are only suggestions. There are no guidelines for this type of URN in this document:

http://example.com/ns/example-interfaces

http://example.com/ns/example-system

4.9. Top-Level Data Definitions

The top-level data organization SHOULD be considered carefully, in advance. Data model designers need to consider how the functionality for a given protocol or protocol family will grow over time.

The separation of configuration data and operational state SHOULD be considered carefully. It is often useful to define separate top-level containers for configuration and non-configuration data. There SHOULD only be one top-level data node defined in each YANG module for all configuration data nodes, if any configuration data nodes are defined at all. There MAY be one top-level data node defined in each YANG module for all non-configuration data nodes, if any non-configuration data nodes are defined at all.

The names and data organization SHOULD reflect persistent information, such as the name of a protocol. The name of the working group SHOULD NOT be used because this may change over time.

A mandatory database data definition is defined as a node that a client must provide for the database to be valid. The server is not required to provide a value.

Top-level database data definitions MUST NOT be mandatory. If a mandatory node appears at the top level, it will immediately cause the database to be invalid. This can occur when the server boots or when a module is loaded dynamically at runtime.

4.10. Data Types

Selection of an appropriate data type (i.e., built-in type, existing derived type, or new derived type) is very subjective, and therefore few requirements can be specified on that subject.

Data model designers SHOULD use the most appropriate built-in data type for the particular application.

If extensibility of enumerated values is required, then the 'identityref' data type SHOULD be used instead of an enumeration or other built-in type.

For string data types, if a machine-readable pattern can be defined for the desired semantics, then one or more pattern statements SHOULD be present.

For string data types, if the length of the string is required to be bounded in all implementations, then a length statement MUST be present.

For numeric data types, if the values allowed by the intended semantics are different than those allowed by the unbounded intrinsic data type (e.g., 'int32'), then a range statement SHOULD be present.

The signed numeric data types (i.e., 'int8', 'int16', 'int32', and 'int64') SHOULD NOT be used unless negative values are allowed for the desired semantics.

For 'enumeration' or 'bits' data types, the semantics for each 'enum' or 'bit' SHOULD be documented. A separate description statement (within each 'enum' or 'bit' statement) SHOULD be present.

4.11. Reusable Type Definitions

If an appropriate derived type exists in any standard module, such as [RFC6991], then it SHOULD be used instead of defining a new derived type.

If an appropriate units identifier can be associated with the desired semantics, then a units statement SHOULD be present.

If an appropriate default value can be associated with the desired semantics, then a default statement SHOULD be present.

If a significant number of derived types are defined, and it is anticipated that these data types will be reused by multiple modules, then these derived types SHOULD be contained in a separate module or

submodule, to allow easier reuse without unnecessary coupling.

The description statement MUST be present.

If the type definition semantics are defined in an external document (other than another YANG module indicated by an import statement), then the reference statement MUST be present.

4.12. Data Definitions

The description statement MUST be present in the following YANG statements:

- anyxml
- augment
- choice
- container
- extension
- feature
- grouping
- identity
- leaf
- leaf-list
- list
- notification
- rpc
- typedef

If the data definition semantics are defined in an external document, (other than another YANG module indicated by an import statement), then a reference statement MUST be present.

The 'anyxml' construct may be useful to represent an HTML banner containing markup elements, such as '' and '', and MAY be used in such cases. However, this construct SHOULD NOT be

used if other YANG data node types can be used instead to represent the desired syntax and semantics.

If there are referential integrity constraints associated with the desired semantics that can be represented with XPath, then one or more 'must' statements SHOULD be present.

For list and leaf-list data definitions, if the number of possible instances is required to be bounded for all implementations, then the max-elements statements SHOULD be present.

If any 'must' or 'when' statements are used within the data definition, then the data definition description statement SHOULD describe the purpose of each one.

4.13. Operation Definitions

If the operation semantics are defined in an external document (other than another YANG module indicated by an import statement), then a reference statement MUST be present.

If the operation impacts system behavior in some way, it SHOULD be mentioned in the description statement.

If the operation is potentially harmful to system behavior in some way, it MUST be mentioned in the Security Considerations section of the document.

4.14. Notification Definitions

The description statement MUST be present.

If the notification semantics are defined in an external document (other than another YANG module indicated by an import statement), then a reference statement MUST be present.

5. IANA Considerations

This document registers one URI in the IETF XML registry [RFC3688].

The following registration has been made:

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

Registrant Contact: The NETMOD WG of the IETF.

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

Per this document, the following assignment has been made in the YANG Module Names Registry for the YANG module template in Appendix B.

Field	Value
Name	ietf-template
Namespace	urn:ietf:params:xml:ns:yang:ietf-template
Prefix	temp
Reference	RFC XXXX

YANG Registry Assignment

6. Security Considerations

This document defines documentation guidelines for NETCONF content defined with the YANG data modeling language. The guidelines for how to write a Security Considerations section for a YANG module are defined in the online document

http://trac.tools.ietf.org/area/ops/trac/wiki/yang-security-guidelines

This document does not introduce any new or increased security risks into the management system.

The following section contains the security considerations template dated 2010-06-16. Be sure to check the webpage at the URL listed above in case there is a more recent version available.

Each specification that defines one or more YANG modules MUST contain a section that discusses security considerations relevant to those modules. This section MUST be patterned after the latest approved template (available at

http://www.ops.ietf.org/netconf/yang-security-considerations.txt).

In particular, writable data nodes that could be especially disruptive if abused MUST be explicitly listed by name and the associated security risks MUST be spelled out.

Similarly, readable data nodes that contain especially sensitive information or that raise significant privacy concerns MUST be explicitly listed by name and the reasons for the sensitivity/privacy concerns MUST be explained.

Further, if new RPC operations have been defined, then the security considerations of each new RPC operation MUST be explained.

6.1. Security Considerations Section Template

X. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

```
-- if you have any writable data nodes (those are all the
-- "config true" nodes, and remember, that is the default)
-- describe their specific sensitivity or vulnerability.
```

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

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

```
-- for all YANG modules you must evaluate whether any readable data
-- nodes (those are all the "config false" nodes, but also all other
-- nodes, because they can also be read via operations like get or
-- get-config) are sensitive or vulnerable (for instance, if they
-- might reveal customer information or violate personal privacy
-- laws such as those of the European Union if exposed to
-- unauthorized parties)
```

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

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

```
-- if your YANG module has defined any rpc operations
-- describe their specific sensitivity or vulnerability.
```

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

<list RPC operations and state why they are sensitive>

7. Acknowledgments

The structure and contents of this document are adapted from [RFC4181], guidelines for MIB Documents, by C. M. Heard.

The working group thanks Martin Bjorklund, Juergen Schoenwaelder, and Ladislav Lhotka for their extensive reviews and contributions to this document.

8. Changes Since RFC 6087

The following changes have been made to the guidelines published in [RFC6087]:

- Updated NETCONF reference from RFC 4741 to RFC 6241
- Updated NETCONF over SSH citation from RFC 4742 to RFC 6242
- Updated YANG Types reference from RFC 6021 to RFC 6991
- Updated obsolete URLs for IETF resources
- Changed top-level data node guideline
- Clarified XPath usage for a literal value representing a YANG identity
- Clarified XPath usage for a when-stmt
- Added terminology guidelines
- Added YANG tree diagram guideline

9. References

9.1. Normative References

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

[RFC2223] Postel, J. and J. Reynolds, "Instructions to RFC Authors", RFC 2223, October 1997.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5378] Bradner, S. and J. Contreras, "Rights Contributors Provide to the IETF Trust", BCP 78, RFC 5378, November 2008.

[RFC5741] Daigle, L., Kolkman, O., and IAB, "RFC Streams, Headers, and Boilerplates", RFC 5741, December 2009.

[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.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

[W3C.REC-xpath-19991116] 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>.

9.2. Informative References

[RFC-STYLE] Braden, R., Ginoza, S., and A. Hagens, "RFC Document Style", September 2009, <http://www.rfc-editor.org/rfc-style-guide/rfc-style>.

[RFC4181] Heard, C., "Guidelines for Authors and Reviewers of MIB

Documents", BCP 111, RFC 4181, September 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.

Appendix A. Module Review Checklist

This section is adapted from RFC 4181.

The purpose of a YANG module review is to review the YANG module both for technical correctness and for adherence to IETF documentation requirements. The following checklist may be helpful when reviewing an Internet-Draft:

- I-D Boilerplate -- verify that the draft contains the required Internet-Draft boilerplate (see http://www.ietf.org/id-info/guidelines.html), including the appropriate statement to permit publication as an RFC, and that I-D boilerplate does not contain references or section numbers.

- Abstract -- verify that the abstract does not contain references, that it does not have a section number, and that its content follows the guidelines in http://www.ietf.org/id-info/guidelines.html.

- Copyright Notice -- verify that the draft has the appropriate text regarding the rights that document contributers provide to the IETF Trust [RFC5378]. Verify that it contains the full IETF Trust copyright notice at the beginning of the document. The IETF Trust Legal Provisions (TLP) can be found at:

  http://trustee.ietf.org/license-info/

- Security Considerations section -- verify that the draft uses the latest approved template from the OPS area website (http://trac.tools.ietf.org/area/ops/trac/wiki/yang-security-guidelines) and that the guidelines therein have been followed.

- IANA Considerations section -- this section must always be present. For each module within the document, ensure that the IANA Considerations section contains entries for the following IANA registries:

  XML Namespace Registry: Register the YANG module namespace.

  YANG Module Registry: Register the YANG module name, prefix, namespace, and RFC number, according to the rules specified in [RFC6020].

- References -- verify that the references are properly divided between normative and informative references, that RFC 2119 is included as a normative reference if the terminology defined therein is used in the document, that all references required by the boilerplate are present, that all YANG modules containing imported items are cited as normative references, and that all citations point to the most current RFCs unless there is a valid reason to do otherwise (for example, it is OK to include an informative reference to a previous version of a specification to help explain a feature included for backward compatibility). Be sure citations for all imported modules are present somewhere in the document text (outside the YANG module).

- License -- verify that the draft contains the Simplified BSD License in each YANG module or submodule. Some guidelines related to this requirement are described in Section 3.1. Make sure that the correct year is used in all copyright dates. Use the approved text from the latest Trust Legal Provisions (TLP) document, which can be found at:

  http://trustee.ietf.org/license-info/

- Other Issues -- check for any issues mentioned in http://www.ietf.org/id-info/checklist.html that are not covered elsewhere.

- Technical Content -- review the actual technical content for compliance with the guidelines in this document. The use of a YANG module compiler is recommended when checking for syntax errors. A list of freely available tools and other information can be found at:

  http://trac.tools.ietf.org/wg/netconf/trac/wiki

  Checking for correct syntax, however, is only part of the job. It is just as important to actually read the YANG module document from the point of view of a potential implementor. It is particularly important to check that description statements are sufficiently clear and unambiguous to allow interoperable implementations to be created.

Appendix B. YANG Module Template

<CODE BEGINS> file "ietf-template@2010-05-18.yang"

```
module ietf-template {

  // replace this string with a unique namespace URN value
  namespace
    "urn:ietf:params:xml:ns:yang:ietf-template";

  // replace this string, and try to pick a unique prefix
  prefix "temp";


  // import statements here: e.g.,
  // import ietf-yang-types { prefix yang; }
  // import ietf-inet-types { prefix inet; }

  // identify the IETF working group if applicable
  organization
     "IETF NETMOD (NETCONF Data Modeling Language) Working Group";

  // update this contact statement with your info
  contact
     "WG Web:   <http://tools.ietf.org/wg/your-wg-name/>
      WG List:  <mailto:your-wg-name@ietf.org>

      WG Chair: your-WG-chair
                <mailto:your-WG-chair@example.com>

      Editor:   your-name
                <mailto:your-email@example.com>";


  // replace the first sentence in this description statement.
  // replace the copyright notice with the most recent
  // version, if it has been updated since the publication
  // of this document
  description
   "This module defines a template for other YANG modules.

    Copyright (c) <insert year> 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 "RFC XXXX";

        // RFC Ed.: remove this note
        // Note: extracted from RFC XXXX


        // replace '2010-05-18' with the module publication date
        // The format is (year-month-day)
        revision "2010-05-18" {
          description
            "Initial version";
        }

        // extension statements

        // feature statements

        // identity statements

        // typedef statements

        // grouping statements

        // data definition statements

        // augment statements

        // rpc statements

        // notification statements

        // DO NOT put deviation statements in a published module

      }


    <CODE ENDS>
```

Author's Address

Andy Bierman
YumaWorks

Email: andy@yumaworks.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 26, 2014

A. Bierman
YumaWorks

December 23, 2013

YANG Conformance Specification
draft-bierman-netmod-yang-conformance-02

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 June 26, 2014.

Copyright Notice

Table of Contents

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]:

- capability
- client
- datastore
- protocol operation
- server

1.1.2. YANG

The following terms are defined in [RFC6020]:

- data node
- extension
- feature

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

1.1.3. Terms

The following terms are used within this document:

- 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.
- 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).
- 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.
- 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.
- object: a conceptual data structure represented by a YANG data, rpc, or notification statement.
- 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.

- 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.

- 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 in 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 identity 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:

- Provide simple documentation mechanisms that are readable and easy to understand.
- Provide simple mechanisms that can scale in usage from one module to thousands of modules.
- Provide per use-case conformance profiles, which allow multiple conformance levels to be specified for a single module.
- Provide per use-case conformance profiles, which allow multiple modules to be specified for a single conformance profile.
- Clarify the usage relationship between an augmented module and the augmenting module.
- Clarify the usage relationship between modules that represent parts of the same conceptual high-level service.
- Provide the ability to specify a stable conformance definition that cannot implicitly change if YANG modules are updated.
- 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 one or more YANG modules and/or NETCONF protocol capabilities. Each package has one or more conformance profiles that describe the server implementation requirements to conform to a specific profile within a package.

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. A particular revision of a YANG package can be considered a snapshot in time of the specific module and profile revisions required for package conformance.

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 or more use-cases or variants 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 conform to one or more conformance profiles within a YANG package.

3.5. Conformance Profile Capability

A new NETCONF capability URI is defined to advertise YANG package conformance. A server will announce conformance for one or more conformance profiles, 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";
      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 2013-12-23 {
         description "First revision";
         reference "TBD";
      }

      profile base {
        description "Basic requirements for YANG types.";

        require-module ietf-yang-types {
          revision 2013-07-15;
          require-conformance import;
          description
            "Support for YANG types  is required.";
          reference "RFC 6991, section 3.";
        }

        require-module ietf-inet-types {
          revision 2013-07-15;
          require-conformance import;
          description
            "Support for INET types  is required.";
          reference "RFC 6991, section 4.";
        }
      }
    }
```

In this example, 3 different conformance profiles are defined for the YANG package named "ietf-routing-pkg":

- 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.

- ipv4: a server that supports IPv4 routing configuration

- ipv6: a server that supports IPv6 routing configuration

```
     package ietf-routing-pkg {

       namespace "urn:ietf:params:xml:ns:yang:ietf-routing-pkg";
       organization
         "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
       contact " ** WG Chairs ** ";
       description
          "This package defines conformance profiles for IPv4 and IPv6
           routers.";

       revision 2013-12-23 {
          description "First revision";
          reference "TBD";
       }

       profile base {
         description "Base module requirements for routing";

         require-package ietf-types-pkg {
            revision 2013-12-23;
         }

         require-module ietf-routing {
            revision 2013-11-07;
            require-conformance full;
            description "The base routing module is required";
            reference "draft-ietf-netmod-routing-cfg-12.txt";
         }

         require-module ietf-interfaces {
            revision 2013-12-23;
            require-conformance augment;
            description
              "The interface and interface-state tables are augmented.";
            reference "draft-ietf-netmod-interfaces-cfg-15.txt";
         }
       }

       profile ipv4 {
         description "Base module requirements for routing";

         include-profile base;

         require-module ietf-ipv4-unicast-routing {
            revision 2013-11-07;
            require-conformance full;
            description "Require IPv4 unicast routing support";
            reference "draft-ietf-netmod-routing-cfg-12.txt";
```

```
        }
      }

      profile ipv6 {
        description "Base module requirements for routing";

        include-profile base;

        require-module ietf-ipv6-unicast-routing {
           revision 2013-11-07;
           require-conformance full;
           description "Require IPv6 unicast routing support";
           reference "draft-ietf-netmod-routing-cfg-12.txt";
        }
      }
    }
```

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

- core: core NETCONF functionality.

- running: a server that supports writing directly to the running datastore.

- startup: a server that supports writing directly to the running datastore and also has a separate startup datastore.

- candidate:- running: a server that supports writing to the candidate datastore and committing all edits at once to the running datastore.

- 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";
      organization
        "IETF NETMOD (NETCONF Data Modeling Language) Working Group";
      contact " ** WG Chairs ** ";
      description
         "This package defines some conformance profiles for the
          NETCONF protocol.";

      revision 2013-12-23 {
```

```
          description "First revision";
          reference "TBD";
        }

        profile core {
          description
            "Basic requirements for complete NETCONF servers.";

          require-package ietf-types-pkg {
             revision 2013-12-23;
          }

          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 {
          revision 2011-06-01;
          description
            "Support for <with-defaults> RPC parameter is required.";
          reference "RFC 6243, section 5";
        }

        require-module ietf-netconf-acm {
          revision 2012-02-22;
          description
            "Base module implementation of NACM is required.";
          reference "RFC 6536, section 3.5.2";
        }

        require-module ietf-netconf-monitoring {
          revision 2010-10-04;
          description
            "Implementation of NETCONF monitoring is required.";
          reference "RFC 6022, section 5";
        }

        require-module ietf-netconf-notifications {
          revision 2012-02-06;
          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
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
profile	4.3	1..n
reference	RFC 6020, 7.19.4	0..1
revision	RFC 6020, 7.1.9	1..n
yangconf-version	4.2	0..1

4.2. The yangconf-version Statement

The optional "yangconf-version" statement specifies which version of the YANG conformance specification language was used in developing the module. The statement's argument is a string. If present, it MUST contain the value "1", which is the current YANG Conformance language version and the default value.

Handling of the "yangconf-version" statement for versions other than "1" (the version defined here) is out of scope for this specification. Any document that defines a higher version will need to define the backward compatibility of such a higher version.

4.3. 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.

A conformance profile SHOULD NOT require any capabilities, modules, definitions, within a module, that have an obsolete "status" statement value. If a conformance profile does reference obsolete definitions then it SHOULD be republished with an obsolete status, or a new YANG package revision published which updates the obsolete profile definition.

4.3.1. The profile Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
include-profile	4.4	0..n
reference	RFC 6020, 7.19.4	0..1
require-capability	4.12	0..n
require-module	4.5	0..n
require-package	4.10	0..n
status	RFC 6020, 7.19.2	0..1

4.4. 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.

4.5. 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.

4.5.1. The require-module Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
revision	4.6	1
reference	RFC 6020, 7.19.4	0..1
require-conformance	4.7	0..1
require-feature	4.8	0..n
require-object	4.9	0..n

4.6. The revision Statement

The "revision" statement is a mandatory statement used to specify the revision date for the required module needed for compliance to the conformance profile containing this statement.

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 a data-model specific matter whether or not revisions of the required module released after this date are acceptable for profile conformance.

4.7. 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:

- full: Full implementation of the module base is required. This is the default value if the require-conformance statement is not present.

- augment: full implementation of the objects that are augmented in this module (from the augmenting module) is required.

- 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.

- 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.

4.8. 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.

4.8.1. The require-feature Substatements

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

4.8.2. Usage Example

```
require-module ietf-ip {
  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";
  }
}
```

4.9. 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.]

4.9.1. The require-object Substatements

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

4.9.2. Usage Example

```
require-module ietf-system {
  revision 2013-07-4;
  require-conformance ad-hoc;
  require-object /system-state/clock/current-datetime {
    description
      "The current system time must be provided.";
    reference
       "RFC XXXX; Section XXXX";
  }
}
```

4.10. 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.

4.10.1. The require-package Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
revision	4.6	1
reference	RFC 6020, 7.19.4	0..1
require-profile	4.11	0..1

4.10.2. Usage Example

```
require-package ietf-routing-pkg {
  revision 2013-12-23;
  require-profile ipv4;
  description
    "Support for IPv4 routing configuration is required.";
}
```

4.11. 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. The server MUST advertise support for the specified conformance profile name within the "profiles" list parameter in the capability URI for the required package.

4.12. 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.

4.12.1. The require-capability Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
reference	RFC 6020, 7.19.4	0..1
require-parameter	4.13	0..n

4.12.2. 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";
  }
}
```

4.13. The require-parameter Statement

The "require-parameter" statement is used to indicate that the specified URI parameter for the parent NETCONF capability URI is required for profile conformance.

It takes as argument a string identifying the parameter name that is required, and MAY be followed by a block of substatements that describe the NETCONF parameter usage within the conformance profile. If no substatements are present then any value for the parameter is permitted for conformance.

If this parameter appears more than once within a "require-capability" statement then all the specified parameters are required for the server to meet profile conformance requirements.

4.13.1. The require-parameter Substatements

Substatement	Reference	Cardinality
description	RFC 6020, 7.19.3	0..1
reference	RFC 6020, 7.19.4	0..1
require-value	4.14	0..n

4.13.2. Usage Example

```
profile valid-url {
  description
    "A profile for requiring that the scheme parameter
     be present in the :url capability. Any value is
     allowed.";
  require-capability
    "urn:ietf:params:netconf:capability:url:1.0" {
    description
      "Support for the :url capability is required.";
    reference "RFC 6241, section 8.8";
    require-parameter scheme {
      description
        "Support for the scheme parameter is required.";
      reference "RFC 6241, section 8.8.3";
  }
}
```

4.14. The require-value Statement

The "require-value" statement is used to indicate an acceptable value for the parent capability parameter for profile conformance.

It takes as argument a string identifying a parameter value that is required, and MAY be followed by a block of substatements that describe the NETCONF parameter value usage within the conformance profile.

If this parameter appears more than once within a "require-parameter" statement then only one of the values is required to match for the server to meet profile conformance requirements.

4.14.1. The require-value 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 ftp-support {
  description
    "A profile for requiring FTP support for the 'url'
     capability.";
  require-capability
    "urn:ietf:params:netconf:capability:url:1.0" {
    description
      "Support for the :url capability is required.";
    reference "RFC 6241, section 8.8";
    require-parameter scheme {
      description
        "Support for the 'scheme' parameter is required.";
      reference "RFC 6241, section 8.8.3";
      require-value ftp {
        description
          "Support for 'ftp' transfer is required.";
      }
    }
  }
}
```

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:

- An existing conformance profile definition MAY be altered to correct errors in the definition.
- New statements (e.g. new requirements) MAY be added to an existing conformance profile.
- Existing requirements MUST NOT be removed from a conformance profile.
- Existing requirements MUST NOT be altered such that the existing functionality would be taken away from clients.
- 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.
- 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   = "profiles=" profile-list
profile-list        = profile-name *("," profile-name)
```

Where:

- "package-name" is the name of the YANG package
- "revision-date" is the revision date of the YANG package
- "profile-list" is a list of the conformance profiles supported by the server

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=2013-12-23&profiles=ipv4,ipv6
</capability>
```

7. YANG Conformance ABNF

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

```
    package-stmt     = optsep package-keyword sep identifier-arg-str
                       optsep
                       "{" stmtsep
                           [yangconf-version-stmt]
                           namespace-stmt
                           meta-stmts
                           revision-stmts
                           package-body-stmts
                       "}" optsep

    yangconf-version-stmt   = yangconf-version-keyword sep
                              yangconf-version-arg-str
                              optsep stmtend

    yangconf-version-arg-str = < a string that matches the rule
                                 yangconf-version-arg >

    yangconf-version-arg    = "1"

    namespace-stmt      = namespace-keyword sep uri-str
                          optsep stmtend

    package-body-stmts  = 1*(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
                            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
                                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]
                                *(require-parameter-stmt stmtsep)
                            "}")

       require-parameter-stmt = require-parameter-keyword sep
```

```
                        identifier-arg-str optsep
                        (";" /
                         "{" stmtsep
                             ;; these stmts can appear in any order
                             [description-stmt stmtsep]
                             [reference-stmt stmtsep]
                             *(require-value-stmt stmtsep)
                         "}")

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

      value-arg-str = <string containing allowed value>

      revision-stmt = 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'
    conformance-keyword         = 'conformance'
    full-keyword                = 'full'
    include-profile-keyword     = 'include-profile'
    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-parameter-keyword   = 'require-parameter'
    require-profile-keyword     = 'require-profile'
    require-value-keyword       = 'require-value'
    revision-keyword            = 'revision'
    yangconf-version-keyword    = 'yangconf-version'

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

<CODE ENDS>

8. IANA Considerations

TBD

9. Security Considerations

TBD

10. Open Issues

- 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.

- 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?

- What package and profile naming conventions are needed? Do these identifiers need to share the same namespace as YANG modules?

- 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.

- Is a media type for a YANG package needed similar to the definition in RFC 6020, section 14.1?

11. Change Log

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

11.1. 01-02

- removed 'min-revision' and 'max-revision' statements.
- added mandatory revision-stmt for profiles and require-package statements.
- changed package capability to allow the server to advertise multiple profiles for a YANG package instead of 1.
- changed required package contents from 0 to 1.
- removed 'category' statement.
- add 'require-parameter' statement to require-capability-stmt.
- add 'require-value' statement to require-parameter-stmt.
- removed 'prefix-stmt' from YANG package.
- removed 'header-stmts' from ABNF so prefix-stmt would be removed.
- added 'yangconf-version-stmt' to ABNF, cloned from RFC 6020
- added 'namespace-stmt' to ABNF, cloned from RFC 6020
- remove conformance profile 'ip' from example, since this is now achieved by advertising multiple names in the 'profiles' parameter in the 'package' capability URI.

11.2. 00-01

- fixed typos in text and examples
- updated ietf-routing-pkg example
- 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.

Table of Contents

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:

- 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 |
           +-----------------------+-------------+
           | yangxp:xpath-type     | 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: 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

Table of Contents

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:

- A set of filters with a set of matching criteria that a packet must satisfy for the rule to be applied.
- 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.

- 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.

- 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.

- 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 SPFs. 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 SPFs.

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 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:spfs/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. Intead, 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:

- Introduce a new YANG module to extend the interface configuration YANG module.
- Import modules "interfaces" [if-config] (prefix: "if") and "stateless-pf" (prefix: "spf").
- 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 SPFs requires a number of new data types introduced in this data model. Table 1 depicts data types that are unique to SPFs. Table 2 depicts data types that are required by SPFs, 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.

- 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.)
- 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 eleement "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".

- name
- spf-type
- enable-capture-global
- capture-session-id-global
- 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.
- 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-tunnel1</port-group>
</src-port-group-name>

<port-groups>
  <port-group>
    <name>port-tunnel1</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". Eeach 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:

- dscp: Match packet on DSCP value.
- 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:

- protocol
- spf:FILTER-COMMON: see Section 4.3
- fragments: When present, it matches the non-initial fragment.
- 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.
- 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-tunnel1</port-group-name>
```

- dest-ports choice: Analogous to "src-ports".
- 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.

- icmp-type
- icmp-code
- packet-length-or-range choice
- 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>
```

- tcp-flag-mask
- tcp-flag-operation
- 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.

- afi
- 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,
- ipv4-pfes: ipv4-pfe has similar structure to ipv6-pfes.
- 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 ip189.168.0.0 255.255.0.0 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-SOUCE-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 SPFs.";
       }

       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 {
                        // abosolute 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
                        "Operant 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 fiter 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:spfs/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)

                bpdu-sap      BPDU SAP encapsulated packets
                bpdu-snap     BPDU SNAP encapsulated packets
                ipx-arpa      IPX Advanced Research Projects Agency
                             (ARPA)
                ipx-non-arpa  IPX non arpa
                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

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.

Table of Contents

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 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]:

- anyxml
- augment
- container
- data node
- data tree
- datatype
- feature
- identity
- instance identifier
- leaf
- leaf-list
- list
- module
- submodule

The following terms are defined in [XMLNS]:

- local name
- prefixed name
- 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:

- YANG data model consisting of a set of YANG modules,
- type of the input document,
- 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