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

M. Bjorklund
Tail-f Systems
J. Schoenwaelder
Jacobs University
October 18, 2010

snmp cfg
draft-bjorklund-netmod-snmp-cfg-00

Abstract

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

Status of this Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Introduction 3
2. Keywords 4
3. Overview 5
4. snmp 6
5. snmp-common 7
6. snmp-agent 11
7. snmp-community 14
8. snmp-notification 16
9. snmp-target 19
10. snmp-target-params 22
11. snmp-usm 24
12. snmp-vacm 27
13. IANA Considerations 32
14. Security Considerations 33
15. Normative References 34
Appendix A. Example configurations 35
Authors' Addresses 36

1. Introduction

TBD.

2. Keywords

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].

3. Overview

TBD.

4. snmp

<CODE BEGINS> file "snmp.yang"

```
module snmp {
  namespace "http://yang-central.org/ns/snmp";
  prefix "snmp";

  include snmp-common {
    revision-date 2010-10-17;
  }
  include snmp-agent {
    revision-date 2010-10-17;
  }
  include snmp-community {
    revision-date 2010-10-17;
  }
  include snmp-notification {
    revision-date 2010-10-17;
  }
  include snmp-target {
    revision-date 2010-10-17;
  }
  include snmp-target-params {
    revision-date 2010-10-17;
  }
  include snmp-vacm {
    revision-date 2010-10-17;
  }
  include snmp-usm {
    revision-date 2010-10-17;
  }

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

  revision 2010-10-17 {
    description
      "Initial revision.";
  }

}
```

<CODE ENDS>

5. snmp-common

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

```
submodule snmp-common {

  belongs-to snmp {
    prefix snmp;
  }

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

  revision 2010-10-17 {
    description
      "Initial revision.";
  }

  /* Collection of SNMP features */

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

  feature multiple-contexts {
    description
      "A server implements this feature if it supports other contexts
      than the default context.";
  }

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

  /* Collection of SNMP specific data types */

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

```
        "RFC 3411: An Architecture for Describing SNMP Management
                   Frameworks";
    }

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

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

    typedef sec-name {
      type admin-string;
      description
        "The sec-name type represents an SNMP security name.";
    }

    typedef mp-model {
      type union {
        type enumeration {
          enum any { value 0; }
          enum v1  { value 1; }
          enum v2c { value 2; }
          enum v3  { value 3; }
        }
        type int32 {
          range "0..2147483647";
        }
      }
      reference
        "RFC3411: An Architecture for Describing SNMP Management
                  Frameworks";
    }

    typedef sec-model {
      type union {
        type enumeration {
          enum v1  { value 1; }
```

```
          enum v2c { value 2; }
          enum usm { value 3; }
        }
        type int32 {
          range "1..2147483647";
        }
      }
      reference
        "RFC3411: An Architecture for Describing SNMP Management
                  Frameworks";
    }

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

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

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

    typedef wildcard-object-identifier {
```

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

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

  }

  <CODE ENDS>
```

6. snmp-agent

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

```
submodule snmp-agent {

  belongs-to snmp {
    prefix snmp;
  }

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

  include snmp-common;

  revision 2010-10-17 {
    description
      "Initial revision.";
  }

  augment /snmp:snmp {

    container agent {

      description
        "Configuration of the SNMP agent";

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

      // FIXME: support multiple endpoints

      leaf ip {
        type inet:ip-address;
        default "0.0.0.0";
        description
          "The IPv4 or IPv6 address to which the agent listens.";
      }

      leaf udp-port {
```

```
          type inet:port-number;
          default "161";
          description
            "The UDP port to which the agent listens.";
        }

        container version {
          description
            "SNMP version used by the agent";
          leaf v1 {
            type empty;
          }
          leaf v2c {
            type empty;
          }
          leaf v3 {
            type empty;
            must "../../engine-id" {
              error-message
                "when v3 is configured, an engine-id must be set";
            }
          }
        }

        container engine-id {
          presence "Sets the local engine-id.";

          description
            "The local SNMP engine's administratively-assigned unique
             identifier.";
          reference "SNMP-FRAMEWORK-MIB.snmpEngineID";

          leaf enterprise-number {
            type uint32;
            mandatory true;
          }
          choice method {
            mandatory true;
            leaf from-ip {
              type inet:ip-address;
            }
            leaf from-mac-address {
              type yang:mac-address;
            }
            leaf from-text {
              type string {
                length 1..27;
              }
```

```
            }
            leaf other {
              type string {
                pattern "[0-9a-fA-F]{2}(:[0-9a-fA-F]{2}){0,27}";
              }
            }
          }
        }
      }

      container system {

        description
          "System group configuration.";

        leaf contact {
          type admin-string;
          default "";
          reference "SNMPv2-MIB.sysContact";
        }

        leaf name {
          type admin-string;
          default "";
          reference "SNMPv2-MIB.sysName";
        }

        leaf location {
          type admin-string;
          default "";
          reference "SNMPv2-MIB.sysLocation";
        }

      }

    }
  }

  <CODE ENDS>
```

7. snmp-community

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

```
submodule snmp-community {

  belongs-to snmp {
    prefix snmp;
  }

  include snmp-common;
  include snmp-target;

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

  revision 2010-10-17 {
    description
     "Initial revision.";
  }

  augment /snmp:snmp {

    list community {
      key index;

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

      leaf index {
        type snmp:identifier;
        description "Index into the community list.";
        reference "SNMP-COMMUNITY-MIB.snmpCommunityIndex";
      }
      leaf name {
        type string;
        description
          "Use only when the community string is not the same as the
          index.";
        reference "SNMP-COMMUNITY-MIB.snmpCommunityName";
      }
      leaf sec-name {
        type snmp:sec-name;
        description
          "If not set, the value of 'name' is operationally used";
        reference "SNMP-COMMUNITY-MIB.snmpCommunitySecurityName";
```

```
        }
        leaf engine-id {
          if-feature snmp:proxy;
          type snmp:engine-id;
          description
            "If not set, the value of the local SNMP engine is
            operationally used by the device.";
          reference "SNMP-COMMUNITY-MIB.snmpCommunityContextEngineID";
        }
        leaf context {
          if-feature snmp:multiple-contexts;
          type snmp:context-name;
          default "";
          reference "SNMP-COMMUNITY-MIB.snmpCommunityContextName";
        }
        leaf target-tag {
          type leafref {
            path "/snmp/target/tag";
          }
          description
            "Used to limit access for this community to the specified
            targets.";
          reference "SNMP-COMMUNITY-MIB.snmpCommunityTransportTag";
        }
      }
    }
  }
```

<CODE ENDS>

8. snmp-notification

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

```
submodule snmp-notification {

  belongs-to snmp {
    prefix snmp;
  }

  include snmp-common;
  include snmp-target;
  include snmp-target-params;

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

  revision 2010-10-17 {
    description
      "Initial revision.";
  }

  augment /snmp:snmp/snmp:target {
    leaf notify-profile {
      if-feature snmp:notification-filter;
      type leafref {
        path "/snmp/notify-profile/name";
      }
    }
  }

  augment /snmp:snmp {

    list notify {

      key name;

      description
        "Targets that will receive notifications.";
      reference "SNMP-NOTIFY-MIB.snmpNotifyTable";

      leaf name {
        type snmp:identifier;
        description
          "An arbitrary name for the list entry.";
        reference "SNMP-NOTIFY-MIB.snmpNotifyName";
      }
```

```
          leaf tag {
            type leafref {
              path "/snmp/target/tag";
            }
            mandatory true;
            description
              "Target tag, selects a set of notification targets.";
            reference "SNMP-NOTIFY-MIB.snmpNotifyTag";
          }
          leaf type {
            type enumeration {
              enum trap { value 1; }
              enum inform { value 2; }
            }
            must
              '. != inform or '
            + 'not(/snmp/target[tag = current()/../name]'
            + '      /../usm[../engine-id] != '
            + '    /snmp/target[tag = current()/../name]/../usm)' {
              error-message
                "When inform is configured, all v3 targets must have an
                 engine-id configured.";
            }
            default trap;
            description "Defines the notification type to be generated.";
            reference "SNMP-NOTIFY-MIB.snmpNotifyType";
          }
        }

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

          description
            "Notification filter profiles associated with targets.";
          reference "SNMP-NOTIFY-MIB.snmpNotifyFilterProfileTable";

          leaf name {
            type snmp:identifier;
            description "Name of the filter profile";
            reference "SNMP-NOTIFY-MIB.snmpNotifyFilterProfileName";
          }
          list subtree {
            key "oids";

            reference "SNMP-NOTIFY-MIB.snmpNotifyFilterTable";

            leaf oids {
```

```
            type wildcard-object-identifier;
            description
              "A family of subtrees included in this filter.";
            reference "SNMP-NOTIFY-MIB.snmpNotifyFilterSubtree
                       SNMP-NOTIFY-MIB.snmpNotifyFilterMask";
          }

          choice type {
            mandatory true;
            leaf included {
              type empty;
              description
                "The family of subtrees is included in the filter.";
            }
            leaf excluded {
              type empty;
              description
                "The family of subtrees is excluded from the filter.";
            }
            reference "SNMP-NOTIFY-MIB.snmpNotifyFilterType";
          }
        }
      }
    }
  }

  <CODE ENDS>
```

9. snmp-target

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

```
 submodule snmp-target {

   belongs-to snmp {
     prefix snmp;
   }

   import ietf-inet-types {
     prefix inet;
   }

   include snmp-common;
   include snmp-usm;

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

   revision 2010-10-17 {
     description
       "Initial revision.";
   }

   augment /snmp:snmp {

     list target {
       key name;

       description "List of targets.";
       reference "SNMP-TARGET-MIB.snmpTargetAddrTable";

       leaf name {
         type snmp:identifier;
         description
           "Identifies the target.";
         reference "SNMP-TARGET-MIB.snmpTargetAddrName";
       }

       // make a choice here so we can add other transports, or
       // they can augment.

       leaf ip {
         type inet:ip-address;
         mandatory true;
         description "Transport IP address of the target";
```

```
          reference "SNMP-TARGET-MIB.snmpTargetAddrTDomain
                     SNMP-TARGET-MIB.snmpTargetAddrTAddress";
        }
        leaf udp-port {
          type inet:port-number;
          default 162;
          description "UDP port number";
          reference "SNMP-TARGET-MIB.snmpTargetAddrTDomain
                     SNMP-TARGET-MIB.snmpTargetAddrTAddress";
        }
        leaf-list tag {
          type snmp:identifier;
          description
            "List of tag values used to select target address.";
          reference "SNMP-TARGET-MIB.snmpTargetAddrTagList";
        }

        leaf timeout {
          type uint32;
          units "0.01 seconds";
          default 1500;
          description
            "Needed only if this target can receive v3 informs.";
          reference "SNMP-TARGET-MIB.snmpTargetAddrTimeout";
        }
        leaf retries {
          type uint8;
          default 3;
          description
            "Needed only if this target can receive v3 informs.";
          reference "SNMP-TARGET-MIB.snmpTargetAddrRetryCount";
        }
        leaf engine-id {
          type leafref {
            path "/snmp/usm/remote/engine-id";
          }
          must '../usm/user-name' {
            error-message
              "When engine-id is set, usm/user-name must also be set.";
          }
          must '/snmp/usm/remote[engine-id=current()]/'
             + 'user[name=current()/../usm/user-name]' {
            error-message
              "When engine-id is set, the usm/user-name must exist in the
              /snmp/usm/remote list for this engine-id.";
          }
          description
            "Needed only if this target can receive v3 informs.
```

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

    }
  }
```

<CODE ENDS>

10. snmp-target-params

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

```
submodule snmp-target-params {

  belongs-to snmp {
    prefix snmp;
  }

  include snmp-common;
  include snmp-community;
  include snmp-target;

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

  revision 2010-10-17 {
    description
      "Initial revision.";
  }

  augment /snmp:snmp/snmp:target {

    /* By including the params directly in the target entry we
       lose some flexibility, but we get a simpler model with less
       cross-references.  In SNMP, two addrEntries can point to the
       same paramsEntry.
    */
    choice params {
      mandatory true;
      reference "SNMP-TARGET-MIB.snmpTargetParamsTable";
      container v1 {
        description "SNMPv1 parameters type";
        // mp-model is v1, sec-level is noAuthNoPriv
        leaf community {
          type leafref {
            path "/snmp/community/index";
          }
          mandatory true;
          reference "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
        }
      }
      container v2c {
        description "SNMPv2 community parameters type";
        // mp-model is v2c, sec-level is noAuthNoPriv
        leaf community {
```

```
            type leafref {
              path "/snmp/community/index";
            }
            mandatory true;
            reference "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
          }
        }
        container usm {
          description "User based SNMPv3 parameters type";
          // mp-model is v3
          leaf user-name {
            type leafref {
              path "/snmp/usm/local/user/name";
            }
            mandatory true;
            reference "SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
          }
          leaf sec-level {
            type sec-level;
            mandatory true;
            reference "SNMP-TARGET-MIB.snmpTargetParamsSecurityLevel";
          }
        }
      }
    }

  }

  <CODE ENDS>
```

11. snmp-usm

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

```
 submodule snmp-usm {

   belongs-to snmp {
     prefix snmp;
   }

   include snmp-common;

   description
     "This submodule contains a collection of YANG definitions for
     configuring the User-based Security Model (USM) of SNMP.";
   reference
     "RFC3414: User-based Security Model (USM) for version 3 of the
               Simple Network Management Protocol (SNMPv3).";

   revision 2010-10-17 {
     description
       "Initial revision.";
   }

   grouping key {
     choice key-type {
       leaf password {
         /* This must be stored in the config; it cannot be derived from
            the SNMP table.  Also, if SNMP is used to set the key,
            this password will not be used anymore */
         type string;
         description
           "Will be used to create a localized key.";
       }
       leaf key {
         type string {
           pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2})*';
         }
         description
           "Authentication key specified as a list of colon-specified
           hexa-decimal octets";
       }
     }
   }

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

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

      leaf name {
        type snmp:identifier;
        reference "SNMP-USER-BASED-SM-MIB.usmUserName
                   SNMP-USER-BASED-SM-MIB.usmUserSecurityName";
      }
      leaf security-name {
        type snmp:identifier;
        description
          "If not set, the value of 'name' is operationally used";
        reference "SNMP-USER-BASED-SM-MIB.usmUserSecurityName";
      }
      container auth {
        presence "enables authentication";
        description "Enables authentication protocol of the user";
        choice protocol {
          mandatory true;
          reference "SNMP-USER-BASED-SM-MIB.usmUserAuthProtocol";
          container md5 {
            presence "md5";
            uses key;
          }
          container sha {
            presence "sha";
            uses key;
          }
        }
      }
      container priv {
        must "../auth" {
          error-message
            "when privacy is used, authentication must also be used";
        }
        presence "enables encryption";
        description
          "Enables encryption for the authentication process.";

        choice protocol {
          mandatory true;
          reference "SNMP-USER-BASED-SM-MIB.usmUserPrivProtocol";
          container des {
            presence "des";
            uses key;
          }
          container aes {
            presence "aes";
            uses key;
```

```
          }
        }
      }
    }
  }

  augment /snmp:snmp {

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

      list remote {
        key "engine-id";

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

        uses user-list;
      }
    }
  }
}

  <CODE ENDS>
```

12. snmp-vacm

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

```
 submodule snmp-vacm {

   belongs-to snmp {
     prefix snmp;
   }

   include snmp-common;

   description
     "This submodule contains a collection of YANG definitions for
     configuring the View-based Access Control Model (VACM) of SNMP.";
   reference
     "RFC3415: View-based Access Control Model (VACM) for the
               Simple Network Management Protocol (SNMP)";

   revision 2010-10-17 {
     description
      "Initial revision.";
   }

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

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

   augment /snmp:snmp {

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

       list group {
         key name;
         description
           "VACM Groups";
         reference "SNMP-VIEW-BASED-ACM-MIB.vacmSecurityToGroupTable";
```

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

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

              A certain combination of sec-name and sec-model MUST NOT
              be mapped to more than one group.";

            leaf sec-name {
              type snmp:sec-name;
              description
                "The securityName of a group member.";
            }

            leaf-list sec-model {
              type snmp:sec-model;
              min-elements 1;
              description
                "The security models under which this securityName
                 is a member of this group.";
            }
          }

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

            leaf context {
 // FIXME: since this is part of the key, it must not have an if-feature
 //           if-feature snmp:multiple-contexts;
              type snmp:context-name;
              description
                "The context (prefix) under which the access rights
                 apply.";
              reference
                "SNMP-VIEW-BASED-ACM-MIB.vacmAccessContextPrefix";
            }
```

```
            leaf context-match {
              if-feature snmp:multiple-contexts;
              type enumeration {
                enum exact;
                enum prefix;
              }
              default exact;
              reference
                "SNMP-VIEW-BASED-ACM-MIB.vacmAccessContextMatch";
            }

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

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

            leaf read-view {
              type leafref {
                path "/snmp/vacm/view/name";
              }
              description
                "The name of the MIB view of the SNMP context authorizing
                 read access.";
              reference
                "SNMP-VIEW-BASED-ACM-MIB.vacmAccessReadViewName";
            }

            leaf write-view {
              type leafref {
                path "/snmp/vacm/view/name";
              }
              description
                "The name of the MIB view of the SNMP context authorizing
                 write access.";
              reference
```

```
            "SNMP-VIEW-BASED-ACM-MIB.vacmAccessWriteViewName";
        }

        leaf notify-view {
          type leafref {
            path "/snmp/vacm/view/name";
          }
          description
            "The name of the MIB view of the SNMP context authorizing
             notify access.";
          reference
            "SNMP-VIEW-BASED-ACM-MIB.vacmAccessNotifyViewName";
        }
      }
    }

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

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

      list subtree {
        key "oids";
        reference
          "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilySubtree";

        leaf oids {
          type snmp:wildcard-object-identifier;
          description
            "A family of subtrees included in this MIB view.";
          reference
            "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilySubtree
             SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyMask";
        }

        choice type {
          mandatory true;
          reference "SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyType";
```

```
              leaf included {
                type empty;
                description
                  "The family of subtrees is included in the MIB view";
              }
              leaf excluded {
                type empty;
                description
                  "The family of subtrees is excluded from the MIB view";
              }
            }
          }
        }
      }
    }
  }
```

<CODE ENDS>

13. IANA Considerations

TBD.

14. Security Considerations

TBD.

15. 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.

Appendix A. Example configurations

TBD.

Authors' Addresses

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com

Juergen Schoenwaelder
Jacobs University

Email: j.schoenwaelder@jacobs-university.de