                   A YANG Data Model for SNMP Configuration

Abstract

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

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc7407.

Table of Contents

1. Introduction

   This document defines a YANG [RFC6020] data model for the
   configuration of SNMP engines. The configuration model is consistent
   with the MIB modules defined in [RFC3411], [RFC3412], [RFC3413],
   [RFC3414], [RFC3415], [RFC3417], [RFC3418], [RFC3419], [RFC3584],
   [RFC3826], [RFC5591], [RFC5592], and [RFC6353] but takes advantage of
   YANG's ability to define hierarchical configuration data models.

   The configuration data model in particular has been designed for SNMP
   deployments where SNMP runs in read-only mode and the Network
   Configuration Protocol (NETCONF) is used to configure the SNMP agent.
   Nevertheless, the data model allows implementations that support
   write access both via SNMP and NETCONF in order to interwork with
   SNMP management applications manipulating SNMP agent configuration
   using SNMP. Further details can be found in Section 3.

   The YANG data model focuses on configuration. Operational state
   objects are not explicitly modeled. The operational state of an SNMP
   agent can be accessed either directly via SNMP or, alternatively, via
   NETCONF using the read-only translation of the relevant SNMP MIB
   modules into YANG modules [RFC6643].

   This document also defines a YANG data model for mapping an X.509
   certificate to a name.

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

2. Data Model

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

2.1. Tree Diagrams

   A simplified graphical representation of the data model is used in
   this document. The meaning of the symbols in these diagrams is as
   follows:

   o Brackets "[" and "]" enclose list keys.

   o Abbreviations before data node names: "rw" means configuration
      (read-write), and "ro" means state data (read-only).

   o Symbols after data node names: "?" means an optional node, "!"
      means a presence container, and "*" denotes a list and leaf-list.

   o Parentheses enclose choice and case nodes, and case nodes are also
      marked with a colon (":").

   o Ellipsis ("...") stands for contents of subtrees that are not
      shown.

2.2. General Considerations

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

   The persistency models in SNMP and NETCONF are quite different. In
   NETCONF, the persistency is defined by the datastore, whereas in
   SNMP, it is defined either explicitly in the data model or on a row-
   by-row basis using the Textual Convention "StorageType". Thus, in
   the YANG model defined here, the "StorageType" columns are not
   present. For implementation guidelines, see Section 3.

   In SNMP, row creation and deletion are controlled using the Textual
   Convention "RowStatus". In NETCONF, creation and deletion are
   handled by the protocol, not in the data model. Thus, in the YANG
   model defined here, the "RowStatus" columns are not present.

2.3. Common Definitions

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

2.4. Engine Configuration

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

```
    +--rw snmp
       +--rw engine
          +--rw enabled?               boolean
          +--rw listen* [name]
          |  +--rw name    snmp:identifier
          |  +--rw (transport)
          |     +--:(udp)
          |        +--rw udp
          |           +--rw ip      inet:ip-address
          |           +--rw port?   inet:port-number
          +--rw version
          |  +--rw v1?     empty
          |  +--rw v2c?    empty
          |  +--rw v3?     empty
          +--rw engine-id?             snmp:engine-id
          +--rw enable-authen-traps?   boolean
```

   The leaf "/snmp/engine/enabled" can be used to enable/disable an SNMP
   engine.

   The list "/snmp/engine/listen" provides configuration of the
   transport endpoints the engine is listening to. In this submodule,
   SNMP over UDP is defined. The Secure Shell (SSH) Protocol, Transport
   Layer Security (TLS), and Datagram Transport Layer Security (DTLS)
   are also supported, defined in "ietf-snmp-ssh" (Section 2.13) and
   "ietf-snmp-tls" (Section 2.12), respectively. The "transport" choice
   is expected to be augmented for other transports.

   The "/snmp/engine/version" container can be used to enable/disable
   the different message processing models [RFC3411].

2.5. Target Configuration

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

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

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

   The "snmpTargetAddrTDomain" and "snmpTargetAddrTAddress" objects are
   mapped to transport-specific YANG nodes. Each transport is
   configured as a separate case in the "transport" choice. In this
   submodule, SNMP over UDP is defined. TLS and DTLS are also
   supported, defined in "ietf-snmp-tls" (Section 2.12). The
   "transport" choice is expected to be augmented for other transports.

   An entry in the list "/snmp/target-params" corresponds to an
   "snmpTargetParamsEntry". This list contains a choice "params", which
   is augmented by submodules specific to the security model, currently,
   "ietf-snmp-community" (Section 2.8), "ietf-snmp-usm" (Section 2.10),
   and "ietf-snmp-tls" (Section 2.12).

2.6. Notification Configuration

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

```
    +--rw snmp
       +--rw notify* [name]
       │  +--rw name     snmp:identifier
       │  +--rw tag      snmp:identifier
       │  +--rw type?    enumeration
       +--rw notify-filter-profile* [name]
          +--rw name        snmp:identifier
          +--rw include*    snmp:wildcard-object-identifier
          +--rw exclude*    snmp:wildcard-object-identifier
```

   This submodule also augments the "target-params" list defined in the
   "ietf-snmp-target" submodule (Section 2.5) with one leaf:

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

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

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

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

   This submodule defines the feature "notification-filter". A server
   implements this feature if it supports SNMP notification filtering
   [RFC3413].

2.7. Proxy Configuration

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

```
    +--rw snmp
       +--rw proxy* [name]
          +--rw name                  snmp:identifier
          +--rw type                  enumeration
          +--rw context-engine-id     snmp:engine-id
          +--rw context-name?         snmp:context-name
          +--rw target-params-in?     snmp:identifier
          +--rw single-target-out?    snmp:identifier
          +--rw multiple-target-out?  snmp:identifier
```

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

   This submodule defines the feature "proxy". A server implements this
   feature if it can act as an SNMP proxy [RFC3413].

2.8. Community Configuration

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

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

   This submodule also augments the "/snmp/target-params/params" choice
   with nodes for the Community-based Security Model used by SNMPv1 and
   SNMPv2c:

```
    +--rw snmp
       +--rw target-params* [name]
       |  ...
       |  +--rw (params)?
       |     +--:(v1)
       |     |  +--rw v1
       |     |     +--rw security-name    snmp:security-name
       |     +--:(v2c)
       |        +--rw v2c
       |           +--rw security-name    snmp:security-name
       +--rw target* [name]
          +--rw mms?        union
```

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

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

2.9. View-Based Access Control Model Configuration

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

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

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

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

2.10. User-Based Security Model Configuration

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

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

   The "{common user params}" are:

```
    +--rw name     snmp:identifier
    +--rw auth!
    |  +--rw (protocol)
    |     +--:(md5)
    |     |  +--rw md5
    |     |     +-- rw key    yang:hex-string
    |     +--:(sha)
    |        +--rw sha
    |           +-- rw key    yang:hex-string
    +--rw priv!
       +--rw (protocol)
          +--:(des)
          |  +--rw des
          |     +-- rw key    yang:hex-string
          +--:(aes)
             +--rw aes
                +-- rw key    yang:hex-string
```

This submodule also augments the "/snmp/target-params/params" choice
with nodes for the SNMP User-based Security Model.

```
    +--rw snmp
       +--rw target-params* [name]
          ...
          +--rw (params)?
             +--:(usm)
                +--rw usm
                   +--rw user-name         snmp:security-name
                   +--rw security-level    security-level
```

In the MIB, there is a single table with local and remote users,
indexed by the engine ID and user name. In the YANG model, there is
one list of local users and a nested list of remote users.

In the MIB, there are several objects related to changing the
authentication and privacy keys. These objects are not present in
the YANG model. However, the localized key can be changed. This
implies that if the engine ID is changed, all users keys need to be
changed as well.

2.11. Transport Security Model Configuration

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

```
    +--rw snmp
       +--rw tsm
          +--rw use-prefix?   boolean
```

This submodule also augments the "/snmp/target-params/params" choice
with nodes for the SNMP Transport Security Model.

```
    +--rw snmp
       +--rw target-params* [name]
          ...
          +--rw (params)?
             +--:(tsm)
                +--rw tsm
                   +--rw security-name     snmp:security-name
                   +--rw security-level    security-level
```

This submodule defines the feature "tsm". A server implements this
feature if it supports the Transport Security Model (TSM) [RFC5591].

2.12. Transport Layer Security Transport Model Configuration

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

```
    +--rw snmp
       ...
       +--rw target* [name]
       |  ...
       |  +--rw (transport)
       |     ...
       |     +--:(tls)
       |     |  +--rw tls
       |     |     +-- {common (d)tls transport params}
       |     +--:(dtls)
       |        +--rw dtls
       |           +-- {common (d)tls transport params}
       +--rw tlstm
          +--rw cert-to-name* [id]
             +--rw id             uint32
             +--rw fingerprint    x509c2n:tls-fingerprint
             +--rw map-type       identityref
             +--rw name           string
```

   The "{common (d)tls transport params}" are:

```
    +--rw ip?                  inet:host
    +--rw port?                inet:port-number
    +--rw client-fingerprint?  x509c2n:tls-fingerprint
    +--rw server-fingerprint?  x509c2n:tls-fingerprint
    +--rw server-identity?     snmp:admin-string
```

This submodule also augments the "/snmp/engine/listen/transport"
choice with objects for the D(TLS) transport endpoints:

```
+--rw snmp
   +--rw engine
      ...
      +--rw listen* [name]
         ...
         +--rw (transport)
            ...
            +--:(tls)
            |   +--rw tls
            |      +--rw ip      inet:ip-address
            |      +--rw port?   inet:port-number
            +--:(dtls)
               +--rw dtls
                  +--rw ip      inet:ip-address
                  +--rw port?   inet:port-number
```

This submodule defines the feature "tlstm". A server implements this
feature if it supports the Transport Layer Security (TLS) Transport
Model (TLSTM) [RFC6353].

2.13. Secure Shell Transport Model Configuration

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

```
+--rw snmp
   ...
   +--rw target* [name]
      ...
      +--rw (transport)
         ...
         +--:(ssh)
            +--rw ssh
               +--rw ip         inet:host
               +--rw port?      inet:port-number
               +--rw username?  string
```

It also augments the "/snmp/engine/listen/transport" choice with
objects for the SSH transport endpoints:

```
+--rw snmp
   +--rw engine
      ...
      +--rw listen* [name]
         ...
         +--rw (transport)
            ...
            +--:(ssh)
               +--rw ssh
                  +--rw ip          inet:host
                  +--rw port?       inet:port-number
                  +--rw username?   string
```

This submodule defines the feature "sshtm". A server implements this
feature if it supports the Secure Shell Transport Model (SSHTM)
[RFC5592].

3. Implementation Guidelines

This section describes some challenges for implementations that
support both the YANG models defined in this document and either
read-write or read-only SNMP access to the same data, using the
standard MIB modules.

As described in Section 2.2, the persistency models in NETCONF and
SNMP are quite different. This poses a challenge for an
implementation to support both NETCONF and SNMP access to the same
data, in particular if the data is writable over both protocols.
Specifically, the configuration data may exist in some combination of
the three NETCONF configuration datastores, and this data must be
mapped to rows in the SNMP tables, in some SNMP contexts, with proper
values for the StorageType columns.

This problem is not new; it has been handled in many implementations
that support configuration of the SNMP engine over a command line
interface (CLI), which normally have a persistency model similar to
NETCONF.

Since there is not one solution that works for all cases, this
document does not provide a recommended solution. Instead, some of
the challenges involved are described below.

3.1. Supporting read-only SNMP Access

   If a device implements only :writable-running, it is trivial to map
   the contents of "running" to data in the SNMP tables, where all
   instances of the StorageType columns have the value "nonVolatile".

   If a device implements :candidate but not :startup, the
   implementation may choose to not expose the contents of the
   "candidate" datastore over SNMP and map the contents of "running" as
   described above. As an option, the contents of "candidate" might be
   accessible in a separate SNMP context.

   If a device implements :startup, the handling of StorageType becomes
   more difficult. Since the contents of "running" and "startup" might
   differ, data in "running" cannot automatically be mapped to instances
   with StorageType "nonVolatile". If a particular entry exists in
   "running" but not in "startup", its StorageType should be "volatile".
   If a particular entry exists in "startup" but not "running", it
   should not be mapped to an SNMP instance, at least not in the default
   SNMP context.

3.2. Supporting read-write SNMP Access

   If the implementation supports read-write access to data over SNMP,
   and specifically creation of table rows, special attention has to be
   given to the handling of the RowStatus and StorageType columns. The
   problem is to determine which table rows to store in the
   configuration datastores and which configuration datastore is
   appropriate for each row.

   The SNMP tables contain a mix of configured data and operational
   state, and only rows with an "active" RowStatus column should be
   stored in a configuration datastore.

   If a device implements only :writable-running, "active" rows with a
   "nonVolatile" StorageType column can be stored in "running". Rows
   with a "volatile" StorageType column are operational state.

   If a device implements :candidate but not :writable-running, all
   configuration changes typically go through the "candidate", even if
   they are done over SNMP. An implementation might have to perform
   some automatic commit of the "candidate" when data is written over
   SNMP, since there is no explicit "commit" operation in SNMP.

   If a device implements :startup, "nonVolatile" rows cannot just be
   written to "running"; they must also be copied into "startup".
   "volatile" rows may be treated as operational state and not copied to
   any datastore, or they may be copied into "running".

Cooperating SNMP management applications may use spin lock objects
(snmpTargetSpinLock [RFC3413], usmUserSpinLock [RFC3414],
vacmViewSpinLock [RFC3415]) to coordinate concurrent write requests.
Implementations supporting modifications of MIB objects protected by
a spin lock via NETCONF should ensure that the spin lock objects are
properly incremented whenever objects are changed via NETCONF. This
allows cooperating SNMP management applications to discover that
concurrent modifications are taking place.

4. Definitions

4.1. Module 'ietf-x509-cert-to-name'

   This YANG module imports typedefs from [RFC6991].

   <CODE BEGINS> file "ietf-x509-cert-to-name.yang"

   module ietf-x509-cert-to-name {

     namespace "urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name";
     prefix x509c2n;

     import ietf-yang-types {
       prefix yang;
     }

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

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

        WG Chair: Thomas Nadeau
                  <mailto:tnadeau@lucidvision.com>

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

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

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

    description
      "This module contains a collection of YANG definitions for
       extracting a name from an X.509 certificate.

         The algorithm used to extract a name from an X.509 certificate
         was first defined in RFC 6353.

         Copyright (c) 2014 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 7407; see
         the RFC itself for full legal notices.";

     reference
       "RFC 6353: Transport Layer Security (TLS) Transport Model for
          the Simple Network Management Protocol (SNMP)";

     revision 2014-12-10 {
       description
         "Initial revision.";
       reference
         "RFC 7407: A YANG Data Model for SNMP Configuration";

     }

     typedef tls-fingerprint {
       type yang:hex-string {
         pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2}){0,254}';
       }
       description
        "A fingerprint value that can be used to uniquely reference
         other data of potentially arbitrary length.

         A tls-fingerprint value is composed of a 1-octet hashing
         algorithm identifier followed by the fingerprint value. The
         first octet value identifying the hashing algorithm is taken
         from the IANA 'TLS HashAlgorithm Registry' (RFC 5246). The
         remaining octets are filled using the results of the hashing
         algorithm.";
       reference
        "RFC 6353: Transport Layer Security (TLS) Transport Model
           for the Simple Network Management Protocol (SNMP).
           SNMP-TLS-TM-MIB.SnmpTLSFingerprint";
     }

```
      /* Identities */

      identity cert-to-name {
        description
          "Base identity for algorithms to derive a name from a
           certificate.";
      }

      identity specified {
        base cert-to-name;
        description
          "Directly specifies the name to be used for the certificate.
           The value of the leaf 'name' in the cert-to-name list is
           used.";
        reference
          "RFC 6353: Transport Layer Security (TLS) Transport Model
              for the Simple Network Management Protocol (SNMP).
              SNMP-TLS-TM-MIB.snmpTlstmCertSpecified";
      }

      identity san-rfc822-name {
        base cert-to-name;
        description
          "Maps a subjectAltName's rfc822Name to a name.  The local part
           of the rfc822Name is passed unaltered, but the host-part of
           the name must be passed in lowercase.  For example, the
           rfc822Name field FooBar@Example.COM is mapped to name
           FooBar@example.com.";
        reference
          "RFC 6353: Transport Layer Security (TLS) Transport Model
              for the Simple Network Management Protocol (SNMP).
              SNMP-TLS-TM-MIB.snmpTlstmCertSANRFC822Name";
      }

      identity san-dns-name {
        base cert-to-name;
        description
          "Maps a subjectAltName's dNSName to a name after first
           converting it to all lowercase (RFC 5280 does not specify
           converting to lowercase, so this involves an extra step).
           This mapping results in a 1:1 correspondence between
           subjectAltName dNSName values and the name values.";
        reference
          "RFC 6353: Transport Layer Security (TLS) Transport Model
              for the Simple Network Management Protocol (SNMP).
              SNMP-TLS-TM-MIB.snmpTlstmCertSANDNSName";
      }
```

```
      identity san-ip-address {
        base cert-to-name;
        description
          "Maps a subjectAltName's iPAddress to a name by
           transforming the binary-encoded address as follows:

              1) for IPv4, the value is converted into a
                 decimal-dotted quad address (e.g., '192.0.2.1').

              2) for IPv6 addresses, the value is converted into a
                 32-character, all-lowercase hexadecimal string
                 without any colon separators.

           This mapping results in a 1:1 correspondence between
           subjectAltName iPAddress values and the name values.";
        reference
          "RFC 6353: Transport Layer Security (TLS) Transport Model
             for the Simple Network Management Protocol (SNMP).
             SNMP-TLS-TM-MIB.snmpTlstmCertSANIpAddress";
      }

      identity san-any {
        base cert-to-name;
        description
          "Maps any of the following fields using the corresponding
           mapping algorithms:

              +------------+-----------------+
              | Type       | Algorithm       |
              |------------+-----------------|
              | rfc822Name | san-rfc822-name |
              | dNSName    | san-dns-name    |
              | iPAddress  | san-ip-address  |
              +------------+-----------------+

           The first matching subjectAltName value found in the
           certificate of the above types MUST be used when deriving
           the name.  The mapping algorithm specified in the
           'Algorithm' column MUST be used to derive the name.

           This mapping results in a 1:1 correspondence between
           subjectAltName values and name values.  The three sub-mapping
           algorithms produced by this combined algorithm cannot produce
           conflicting results between themselves.";
        reference
          "RFC 6353: Transport Layer Security (TLS) Transport Model
             for the Simple Network Management Protocol (SNMP).
             SNMP-TLS-TM-MIB.snmpTlstmCertSANAny";
```

```
      }

      identity common-name {
        base cert-to-name;
        description
          "Maps a certificate's CommonName to a name after converting
           it to a UTF-8 encoding.  The usage of CommonNames is
           deprecated, and users are encouraged to use subjectAltName
           mapping methods instead.  This mapping results in a 1:1
           correspondence between certificate CommonName values and name
           values.";
        reference
          "RFC 6353: Transport Layer Security (TLS) Transport Model
             for the Simple Network Management Protocol (SNMP).
             SNMP-TLS-TM-MIB.snmpTlstmCertCommonName";
      }

      /*
       * Groupings
       */

      grouping cert-to-name {
        description
          "Defines nodes for mapping certificates to names.  Modules
           that use this grouping should describe how the resulting
           name is used.";

        list cert-to-name {
          key id;
          description
            "This list defines how certificates are mapped to names.
             The name is derived by considering each cert-to-name
             list entry in order.  The cert-to-name entry's fingerprint
             determines whether the list entry is a match:

              1) If the cert-to-name list entry's fingerprint value
                 matches that of the presented certificate, then consider
                 the list entry a successful match.

              2) If the cert-to-name list entry's fingerprint value
                 matches that of a locally held copy of a trusted CA
                 certificate, and that CA certificate was part of the CA
                 certificate chain to the presented certificate, then
                 consider the list entry a successful match.

             Once a matching cert-to-name list entry has been found, the
             map-type is used to determine how the name associated with
             the certificate should be determined.  See the map-type
```

          leaf's description for details on determining the name value.
          If it is impossible to determine a name from the cert-to-name
          list entry's data combined with the data presented in the
          certificate, then additional cert-to-name list entries MUST
          be searched to look for another potential match.

          Security administrators are encouraged to make use of
          certificates with subjectAltName fields that can be mapped to
          names so that a single root CA certificate can allow all
          child certificates' subjectAltName fields to map directly to
          a name via a 1:1 transformation.";
        reference
         "RFC 6353: Transport Layer Security (TLS) Transport Model
            for the Simple Network Management Protocol (SNMP).
            SNMP-TLS-TM-MIB.snmpTlstmCertToTSNEntry";

        leaf id {
          type uint32;
          description
            "The id specifies the order in which the entries in the
             cert-to-name list are searched. Entries with lower
             numbers are searched first.";
          reference
            "RFC 6353: Transport Layer Security (TLS) Transport Model
               for the Simple Network Management Protocol
               (SNMP).
               SNMP-TLS-TM-MIB.snmpTlstmCertToTSNID";
        }

        leaf fingerprint {
          type x509c2n:tls-fingerprint;
          mandatory true;
          description
            "Specifies a value with which the fingerprint of the
             full certificate presented by the peer is compared. If
             the fingerprint of the full certificate presented by the
             peer does not match the fingerprint configured, then the
             entry is skipped, and the search for a match continues.";
          reference
            "RFC 6353: Transport Layer Security (TLS) Transport Model
               for the Simple Network Management Protocol
               (SNMP).
               SNMP-TLS-TM-MIB.snmpTlstmCertToTSNFingerprint";
        }

        leaf map-type {
          type identityref {
            base cert-to-name;

```
            }
            mandatory true;
            description
              "Specifies the algorithm used to map the certificate
               presented by the peer to a name.

               Mappings that need additional configuration objects should
               use the 'when' statement to make them conditional based on
               the map-type.";
            reference
              "RFC 6353: Transport Layer Security (TLS) Transport Model
                  for the Simple Network Management Protocol
                  (SNMP).
                  SNMP-TLS-TM-MIB.snmpTlstmCertToTSNMapType";
          }

          leaf name {
            when "../map-type = 'x509c2n:specified'";
            type string;
            mandatory true;
            description
              "Directly specifies the NETCONF username when the
               map-type is 'specified'.";
            reference
              "RFC 6353: Transport Layer Security (TLS) Transport Model
                  for the Simple Network Management Protocol
                  (SNMP).
                  SNMP-TLS-TM-MIB.snmpTlstmCertToTSNData";
          }
        }
      }
    }
  }

  <CODE ENDS>
```

4.2. Module 'ietf-snmp'

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

  module ietf-snmp {

    namespace "urn:ietf:params:xml:ns:yang:ietf-snmp";
    prefix snmp;

    include ietf-snmp-common {
      revision-date 2014-12-10;
    }
    include ietf-snmp-engine {
```

```
     revision-date 2014-12-10;
   }
   include ietf-snmp-target {
     revision-date 2014-12-10;
   }
   include ietf-snmp-notification {
     revision-date 2014-12-10;
   }
   include ietf-snmp-proxy {
     revision-date 2014-12-10;
   }
   include ietf-snmp-community {
     revision-date 2014-12-10;
   }
   include ietf-snmp-usm {
     revision-date 2014-12-10;
   }
   include ietf-snmp-tsm {
     revision-date 2014-12-10;
   }
   include ietf-snmp-vacm {
     revision-date 2014-12-10;
   }
   include ietf-snmp-tls {
     revision-date 2014-12-10;
   }
   include ietf-snmp-ssh {
     revision-date 2014-12-10;
   }

   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: Thomas Nadeau
                <mailto:tnadeau@lucidvision.com>

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

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

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

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

      Copyright (c) 2014 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 7407; see
      the RFC itself for full legal notices.";

   revision 2014-12-10 {
     description
       "Initial revision.";
     reference
       "RFC 7407: A YANG Data Model for SNMP Configuration";
   }

 }

 <CODE ENDS>
```

4.3. Submodule 'ietf-snmp-common'

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

   submodule ietf-snmp-common {

     belongs-to ietf-snmp {
       prefix snmp;
     }

     import ietf-yang-types {
       prefix yang;
     }

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

     contact
       "WG Web:   <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>
```

           WG Chair: Thomas Nadeau
                     <mailto:tnadeau@lucidvision.com>

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

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

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

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

          Copyright (c) 2014 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 7407; see
          the RFC itself for full legal notices.";

       revision 2014-12-10 {
         description
           "Initial revision.";
         reference
           "RFC 7407: A YANG Data Model for SNMP Configuration";
       }

       /* Collection of SNMP-specific data types */

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

            Note that the size of an SnmpAdminString is measured in
            octets, not characters.";

```
       reference
         "RFC 3411: An Architecture for Describing Simple Network
            Management Protocol (SNMP) Management Frameworks.
            SNMP-FRAMEWORK-MIB.SnmpAdminString";
     }

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

     typedef context-name {
       type admin-string {
         length "0..32";
       }
       description
         "The context type represents an SNMP context name.";
       reference
         "RFC 3411: An Architecture for Describing Simple Network
            Management Protocol (SNMP) Management Frameworks";
     }

     typedef security-name {
       type admin-string {
         length "1..32";
       }
       description
         "The security-name type represents an SNMP security name.";
       reference
         "RFC 3411: An Architecture for Describing Simple Network
            Management Protocol (SNMP) Management Frameworks";
     }

     typedef security-model {
       type union {
         type enumeration {
           enum v1  { value 1; }
           enum v2c { value 2; }
           enum usm { value 3; }
           enum tsm { value 4; }
         }
         type int32 {
           range "1..2147483647";
         }
```

```
      }
      reference
        "RFC 3411: An Architecture for Describing Simple Network
           Management Protocol (SNMP) Management Frameworks";
    }

    typedef security-model-or-any {
      type union {
        type enumeration {
          enum any { value 0; }
        }
        type security-model;
      }
      reference
        "RFC 3411: An Architecture for Describing Simple Network
           Management Protocol (SNMP) Management Frameworks";
    }

    typedef security-level {
      type enumeration {
        enum no-auth-no-priv { value 1; }
        enum auth-no-priv    { value 2; }
        enum auth-priv       { value 3; }
      }
      reference
        "RFC 3411: An Architecture for Describing Simple Network
           Management Protocol (SNMP) Management Frameworks";
    }

    typedef engine-id {
      type yang:hex-string {
        pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2}){4,31}';
      }
      description
        "The engine ID specified as a list of colon-specified
        hexadecimal octets, e.g., '80:00:02:b8:04:61:62:63'.";
      reference
        "RFC 3411: An Architecture for Describing Simple Network
           Management Protocol (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 an asterisk
        ('*').";
```

```
      }

      typedef tag-value {
        type string {
          length "0..255";
        }
        description
          "Represents SnmpTagValue as defined in RFC 3413.

          Note that the size of an SnmpTagValue is measured in
          octets, not characters.";
        reference
          "RFC 3413: Simple Network Management Protocol (SNMP)
             Applications.
             SNMP-TARGET-MIB.SnmpTagValue";
      }

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

    }

    <CODE ENDS>
```

4.4. Submodule 'ietf-snmp-engine'

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

    submodule ietf-snmp-engine {

      belongs-to ietf-snmp {
        prefix snmp;
      }

      import ietf-inet-types {
        prefix inet;
      }

      include ietf-snmp-common;

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

      contact
        "WG Web:    <http://tools.ietf.org/wg/netmod/>
```

```
     WG List:  <mailto:netmod@ietf.org>

     WG Chair: Thomas Nadeau
               <mailto:tnadeau@lucidvision.com>

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

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

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

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

     Copyright (c) 2014 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 7407; see
     the RFC itself for full legal notices.";

  revision 2014-12-10 {
    description
      "Initial revision.";
    reference
      "RFC 7407: A YANG Data Model for SNMP Configuration";
  }

  augment /snmp:snmp {

    container engine {

      description
        "Configuration of the SNMP engine.";

      leaf enabled {
        type boolean;
        default "false";
        description
```

```
                    "Enables the SNMP engine.";
            }

            list listen {
              key "name";
              description
                "Configuration of the transport endpoints on which the
                 engine listens.";

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

              choice transport {
                mandatory true;
                description
                  "The transport-protocol-specific parameters for this
                   endpoint.  Submodules providing configuration for
                   additional transports are expected to augment this
                   choice.";
                case udp {
                  container udp {
                    leaf ip {
                      type inet:ip-address;
                      mandatory true;
                      description
                        "The IPv4 or IPv6 address on which the engine
                         listens.";
                    }
                    leaf port {
                      type inet:port-number;
                      description
                        "The UDP port on which the engine listens.

                         If the port is not configured, an engine that
                         acts as a Command Responder uses port 161, and
                         an engine that acts as a Notification Receiver
                         uses port 162.";
                    }
                  }
                }
              }
            }
```

```
         container version {
           description
             "SNMP version used by the engine.";
           leaf v1 {
             type empty;
           }
           leaf v2c {
             type empty;
           }
           leaf v3 {
             type empty;
           }
         }

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

              If this leaf is not set, the device automatically
              calculates an engine ID, as described in RFC 3411.  A
              server MAY initialize this leaf with the automatically
              created value.";
           reference
             "RFC 3411: An Architecture for Describing Simple Network
                Management Protocol (SNMP) Management
                Frameworks.
                SNMP-FRAMEWORK-MIB.snmpEngineID";
         }

         leaf enable-authen-traps {
           type boolean;
           description
             "Indicates whether the SNMP entity is permitted to
              generate authenticationFailure traps.";
           reference
             "RFC 3418: Management Information Base (MIB) for the
                Simple Network Management Protocol (SNMP)
                SNMPv2-MIB.snmpEnableAuthenTraps";
         }
       }
     }
   }

   <CODE ENDS>
```

4.5. Submodule 'ietf-snmp-target'

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

   submodule ietf-snmp-target {

     belongs-to ietf-snmp {
       prefix snmp;
     }

     import ietf-inet-types {
       prefix inet;
     }

     include ietf-snmp-common;

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

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

        WG Chair: Thomas Nadeau
                  <mailto:tnadeau@lucidvision.com>

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

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

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

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

        Copyright (c) 2014 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 7407; see
      the RFC itself for full legal notices.";

    reference
      "RFC 3413: Simple Network Management Protocol (SNMP)
        Applications";

    revision 2014-12-10 {
      description
        "Initial revision.";
      reference
        "RFC 7407: A YANG Data Model for SNMP Configuration";
    }

    augment /snmp:snmp {

      list target {
        key name;
        description
          "List of targets.";
        reference
          "RFC 3413: Simple Network Management Protocol (SNMP)
            Applications.
            SNMP-TARGET-MIB.snmpTargetAddrTable";

        leaf name {
          type snmp:identifier;
          description
            "Identifies the target.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP)
              Applications.
              SNMP-TARGET-MIB.snmpTargetAddrName";
        }
        choice transport {
          mandatory true;
          description
            "Transport address of the target.

            The snmpTargetAddrTDomain and snmpTargetAddrTAddress
            objects are mapped to transport-specific YANG nodes. Each
            transport is configured as a separate case in this
            choice. Submodules providing configuration for additional
            transports are expected to augment this choice.";

```
              reference
                "RFC 3413: Simple Network Management Protocol (SNMP)
                    Applications.
                 SNMP-TARGET-MIB.snmpTargetAddrTDomain
                 SNMP-TARGET-MIB.snmpTargetAddrTAddress";
            case udp {
              reference
               "RFC 3417: Transport Mappings for the Simple Network
                   Management Protocol (SNMP).
                 SNMPv2-TM.snmpUDPDomain
                RFC 3419: Textual Conventions for Transport Addresses.
                 TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv4
                 TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv4z
                 TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv6
                 TRANSPORT-ADDRESS-MIB.transportDomainUdpIpv6z";
              container udp {
                leaf ip {
                  type inet:ip-address;
                  mandatory true;
                  reference
                    "RFC 3413: Simple Network Management Protocol (SNMP).
                        SNMP-TARGET-MIB.snmpTargetAddrTAddress";
                }
                leaf port {
                  type inet:port-number;
                  default 162;
                  description
                    "UDP port number.";
                  reference
                    "RFC 3413: Simple Network Management Protocol (SNMP).
                        SNMP-TARGET-MIB.snmpTargetAddrTAddress";
                }
                leaf prefix-length {
                  type uint8;
                  description
                    "The value of this leaf must match the value of
                     ../snmp:ip.  If ../snmp:ip contains an IPv4 address,
                     this leaf must be less than or equal to 32.  If it
                     contains an IPv6 address, it must be less than or
                     equal to 128.

                     Note that the prefix-length is currently only used
                     by the Community-based Security Model to filter
                     incoming messages.  Furthermore, the prefix-length
                     filtering does not cover all possible filters
                     supported by the corresponding MIB object.";
```

```
                 reference
                   "RFC 3584: Coexistence between Version 1, Version 2,
                      and Version 3 of the Internet-standard
                      Network Management Framework.
                      SNMP-COMMUNITY-MIB.snmpTargetAddrTMask";
             }
           }
         }
       }
       leaf-list tag {
         type snmp:tag-value;
         description
           "List of tag values used to select target addresses.";
         reference
           "RFC 3413: Simple Network Management Protocol (SNMP).
              Applications.
              SNMP-TARGET-MIB.snmpTargetAddrTagList";
       }
       leaf timeout {
         type uint32;
         units "0.01 seconds";
         default 1500;
         description
           "Needed only if this target can receive
            InformRequest-PDUs.";
         reference
           "RFC 3413: Simple Network Management Protocol (SNMP).
              Applications.
              SNMP-TARGET-MIB.snmpTargetAddrTimeout";
       }
       leaf retries {
         type uint8;
         default 3;
         description
           "Needed only if this target can receive
            InformRequest-PDUs.";
         reference
           "RFC 3413: Simple Network Management Protocol (SNMP).
              Applications.
              SNMP-TARGET-MIB.snmpTargetAddrRetryCount";
       }
       leaf target-params {
         type snmp:identifier;
         mandatory true;
         reference
           "RFC 3413: Simple Network Management Protocol (SNMP).
              Applications.
              SNMP-TARGET-MIB.snmpTargetAddrParams";
```

```
          }
        }

        list target-params {
          key name;
          description
            "List of target parameters.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
                Applications.
                SNMP-TARGET-MIB.snmpTargetParamsTable";

          leaf name {
            type snmp:identifier;
          }
          choice params {
            description
              "This choice is augmented with case nodes containing
               configuration parameters specific to the security model.";
          }
        }
      }
    }
  }

  <CODE ENDS>
```

4.6. Submodule 'ietf-snmp-notification'

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

  submodule ietf-snmp-notification {

    belongs-to ietf-snmp {
      prefix snmp;
    }

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

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

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

       WG Chair: Thomas Nadeau
                 <mailto:tnadeau@lucidvision.com>
```

```
      WG Chair: Juergen Schoenwaelder
                <mailto:j.schoenwaelder@jacobs-university.de>

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

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

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

      Copyright (c) 2014 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 7407; see
      the RFC itself for full legal notices.";

   reference
     "RFC 3413: Simple Network Management Protocol (SNMP)
        Applications";

   revision 2014-12-10 {
     description
       "Initial revision.";
     reference
       "RFC 7407: A YANG Data Model for SNMP Configuration";
   }

   feature notification-filter {
     description
       "A server implements this feature if it supports SNMP
       notification filtering.";
     reference
       "RFC 3413: Simple Network Management Protocol (SNMP)
         Applications";
   }

   augment /snmp:snmp {
```

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

           Entries in this list are mapped 1-1 to entries in
           snmpNotifyTable, except that if an entry in snmpNotifyTable
           has an snmpNotifyTag for which no snmpTargetAddrEntry
           exists, then the snmpNotifyTable entry is not mapped to an
           entry in this list.";
        reference
          "RFC 3413: Simple Network Management Protocol (SNMP).
             Applications.
             SNMP-NOTIFICATION-MIB.snmpNotifyTable";

        leaf name {
          type snmp:identifier;
          description
            "An arbitrary name for the list entry.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
               Applications.
               SNMP-NOTIFICATION-MIB.snmpNotifyName";
        }
        leaf tag {
          type snmp:tag-value;
          mandatory true;
          description
            "Target tag, selects a set of notification targets.

             Implementations MAY restrict the values of this leaf
             to be one of the available values of /snmp/target/tag in
             a valid configuration.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
               Applications.
               SNMP-NOTIFICATION-MIB.snmpNotifyTag";
        }
        leaf type {
          type enumeration {
            enum trap { value 1; }
            enum inform { value 2; }
          }
          default trap;
          description
            "Defines the notification type to be generated.";
```

```
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
                Applications.
                SNMP-NOTIFICATION-MIB.snmpNotifyType";
        }
      }

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

        description
          "Notification filter profiles.

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

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

            If an entry in this list is not referred to by any
            /snmp/target/notify-filter-profile, the entry is not mapped
            to snmpNotifyFilterProfileTable.";
        reference
          "RFC 3413: Simple Network Management Protocol (SNMP).
              Applications.
              SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileTable
              SNMP-NOTIFICATION-MIB.snmpNotifyFilterTable";

        leaf name {
          type snmp:identifier;
          description
            "Name of the filter profile.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
                Applications.
                SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileName";
        }

        leaf-list include {
          type snmp:wildcard-object-identifier;
          description
            "A family of subtrees included in this filter.";
```

```
                reference
                  "RFC 3413: Simple Network Management Protocol (SNMP).
                      Applications.
                      SNMP-NOTIFICATION-MIB.snmpNotifyFilterSubtree
                      SNMP-NOTIFICATION-MIB.snmpNotifyFilterMask
                      SNMP-NOTIFICATION-MIB.snmpNotifyFilterType";
            }

            leaf-list exclude {
              type snmp:wildcard-object-identifier;
              description
                "A family of subtrees excluded from this filter.";
              reference
                "RFC 3413: Simple Network Management Protocol (SNMP).
                    Applications.
                    SNMP-NOTIFICATION-MIB.snmpNotifyFilterSubtree
                    SNMP-NOTIFICATION-MIB.snmpNotifyFilterMask
                    SNMP-NOTIFICATION-MIB.snmpNotifyFilterType";
            }
          }

        }

      augment /snmp:snmp/snmp:target-params {
        reference
          "RFC 3413: Simple Network Management Protocol (SNMP).
             Applications.
             SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileTable";
        leaf notify-filter-profile {
          if-feature snmp:notification-filter;
          type leafref {
            path "/snmp/notify-filter-profile/name";
          }
          description
            "This leafref leaf is used to represent the sparse
             relationship between the /snmp/target-params list and the
             /snmp/notify-filter-profile list.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
               Applications.
               SNMP-NOTIFICATION-MIB.snmpNotifyFilterProfileName";
        }
      }

    }

    <CODE ENDS>
```

4.7. Submodule 'ietf-snmp-proxy'

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

   submodule ietf-snmp-proxy {

     belongs-to ietf-snmp {
       prefix snmp;
     }

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

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

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

        WG Chair: Thomas Nadeau
                  <mailto:tnadeau@lucidvision.com>

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

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

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

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

        Copyright (c) 2014 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 7407; see
        the RFC itself for full legal notices.";

```
        reference
          "RFC 3413: Simple Network Management Protocol (SNMP)
            Applications";

      revision 2014-12-10 {
        description
          "Initial revision.";
        reference
          "RFC 7407: A YANG Data Model for SNMP Configuration";
      }

      feature proxy {
        description
          "A server implements this feature if it can act as an
          SNMP proxy.";
        reference
          "RFC 3413: Simple Network Management Protocol (SNMP)
            Applications";
      }

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

        list proxy {
          key name;

          description
            "List of proxy parameters.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
                Applications.
                SNMP-PROXY-MIB.snmpProxyTable";

          leaf name {
            type snmp:identifier;
            description
              "Identifies the proxy parameter entry.";
            reference
              "RFC 3413: Simple Network Management Protocol (SNMP).
                  Applications.
                  SNMP-PROXY-MIB.snmpProxyName";
          }
          leaf type {
            type enumeration {
              enum read { value 1; }
              enum write { value 2; }
              enum trap { value 3; }
              enum inform { value 4; }
```

```
          }
          mandatory true;
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
               Applications.
               SNMP-PROXY-MIB.snmpProxyType";
        }
        leaf context-engine-id {
          type snmp:engine-id;
          mandatory true;
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
               Applications.
               SNMP-PROXY-MIB.snmpProxyContextEngineID";
        }
        leaf context-name {
          type snmp:context-name;
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
               Applications.
               SNMP-PROXY-MIB.snmpProxyContextName";
        }
        leaf target-params-in {
          type snmp:identifier;
          description
            "The name of a target parameters list entry.

             Implementations MAY restrict the values of this
             leaf to be one of the available values of
             /snmp/target-params/name in a valid configuration.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
               Applications.
               SNMP-PROXY-MIB.snmpProxyTargetParamsIn";
        }
        leaf single-target-out {
          when "../type = 'read' or ../type = 'write'";
          type snmp:identifier;
          description
            "Implementations MAY restrict the values of this leaf
             to be one of the available values of /snmp/target/name in
             a valid configuration.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
               Applications.
               SNMP-PROXY-MIB.snmpProxySingleTargetOut";
        }
```

```
        leaf multiple-target-out {
          when "../type = 'trap' or ../type = 'inform'";
          type snmp:tag-value;
          description
            "Implementations MAY restrict the values of this leaf
             to be one of the available values of /snmp/target/tag in
             a valid configuration.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
               Applications.
               SNMP-PROXY-MIB.snmpProxyMultipleTargetOut";
        }
      }
    }
  }
```

    <CODE ENDS>

4.8.   Submodule 'ietf-snmp-community'

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

```
    submodule ietf-snmp-community {

      belongs-to ietf-snmp {
        prefix snmp;
      }

      import ietf-netconf-acm {
        prefix nacm;
      }

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

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

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

         WG Chair: Thomas Nadeau
                   <mailto:tnadeau@lucidvision.com>

         WG Chair: Juergen Schoenwaelder
                   <mailto:j.schoenwaelder@jacobs-university.de>
```

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

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

     description
       "This submodule contains a collection of YANG definitions
        for configuring community-based SNMP.

        Copyright (c) 2014 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 7407; see
        the RFC itself for full legal notices.";

     reference
       "RFC 3584: Coexistence between Version 1, Version 2, and
          Version 3 of the Internet-standard Network
          Management Framework";

     revision 2014-12-10 {
       description
         "Initial revision.";
       reference
         "RFC 7407: A YANG Data Model for SNMP Configuration";
     }

     augment /snmp:snmp {

       list community {
         key index;

         description
           "List of communities.";
         reference
           "RFC 3584: Coexistence between Version 1, Version 2,
              and Version 3 of the Internet-standard
              Network Management Framework.
              SNMP-COMMUNITY-MIB.snmpCommunityTable";

```
          leaf index {
            type snmp:identifier;
            description
              "Index into the community list.";
            reference
              "RFC 3584: Coexistence between Version 1, Version 2,
                  and Version 3 of the Internet-standard
                  Network Management Framework.
                  SNMP-COMMUNITY-MIB.snmpCommunityIndex";
          }
          choice name {
            nacm:default-deny-all;
            description
              "The community name, specified as either a string or
               a binary value.  The binary name is used when the
               community name contains characters that are not legal
               in a string.

               If not set, the value of 'security-name' is operationally
               used as the snmpCommunityName.";
            reference
              "RFC 3584: Coexistence between Version 1, Version 2,
                  and Version 3 of the Internet-standard
                  Network Management Framework.
                  SNMP-COMMUNITY-MIB.snmpCommunityName";
            leaf text-name {
              type string;
              description
                "A community name that can be represented as a
                 YANG string.";
            }
            leaf binary-name {
              type binary;
              description
                "A community name represented as a binary value.";
            }
          }
          leaf security-name {
            type snmp:security-name;
            mandatory true;
            nacm:default-deny-all;
            description
              "The snmpCommunitySecurityName of this entry.";
            reference
              "RFC 3584: Coexistence between Version 1, Version 2,
                  and Version 3 of the Internet-standard
                  Network Management Framework.
                  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
            "RFC 3584: Coexistence between Version 1, Version 2,
                and Version 3 of the Internet-standard
                Network Management Framework.
                SNMP-COMMUNITY-MIB.snmpCommunityContextEngineID";
        }
        leaf context {
          type snmp:context-name;
          default "";
          description
            "The context in which management information is accessed
             when using the community string specified by this entry.";
          reference
            "RFC 3584: Coexistence between Version 1, Version 2,
                and Version 3 of the Internet-standard
                Network Management Framework.
                SNMP-COMMUNITY-MIB.snmpCommunityContextName";
        }
        leaf target-tag {
          type snmp:tag-value;
          description
            "Used to limit access for this community to the specified
             targets.

             Implementations MAY restrict the values of this leaf
             to be one of the available values of /snmp/target/tag in
             a valid configuration.";
          reference
            "RFC 3584: Coexistence between Version 1, Version 2,
                and Version 3 of the Internet-standard
                Network Management Framework.
                SNMP-COMMUNITY-MIB.snmpCommunityTransportTag";
        }
      }
    }

    grouping v1-target-params {
      container v1 {
        description
          "SNMPv1 parameters type.
           Represents snmpTargetParamsMPModel '0',
```

```
           snmpTargetParamsSecurityModel '1', and
           snmpTargetParamsSecurityLevel 'noAuthNoPriv'.";
        leaf security-name {
          type snmp:security-name;
          mandatory true;
          description
            "Implementations MAY restrict the values of this leaf
             to be one of the available values of
             /snmp/community/security-name in a valid configuration.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
                Applications.
                SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
        }
      }
    }

    grouping v2c-target-params {
      container v2c {
        description
          "SNMPv2 community parameters type.
           Represents snmpTargetParamsMPModel '1',
           snmpTargetParamsSecurityModel '2', and
           snmpTargetParamsSecurityLevel 'noAuthNoPriv'.";
        leaf security-name {
          type snmp:security-name;
          mandatory true;
          description
            "Implementations MAY restrict the values of this leaf
             to be one of the available values of
             /snmp/community/security-name in a valid configuration.";
          reference
            "RFC 3413: Simple Network Management Protocol (SNMP).
                Applications.
                SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
        }
      }
    }

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

```
      augment /snmp:snmp/snmp:target {
        when "snmp:v1 or snmp:v2c";
        leaf mms {
          type union {
            type enumeration {
              enum "unknown" { value 0; }
            }
            type int32 {
              range "484..max";
            }
          }
          default "484";
          description
            "The maximum message size.";
          reference
            "RFC 3584: Coexistence between Version 1, Version 2,
                and Version 3 of the Internet-standard
                Network Management Framework.
                SNMP-COMMUNITY-MIB.snmpTargetAddrMMS";
        }
      }

    }

    <CODE ENDS>
```

4.9. Submodule 'ietf-snmp-vacm'

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

    submodule ietf-snmp-vacm {

      belongs-to ietf-snmp {
        prefix snmp;
      }

      include ietf-snmp-common;

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

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

         WG Chair: Thomas Nadeau
                   <mailto:tnadeau@lucidvision.com>
```

```
        WG Chair: Juergen Schoenwaelder
                  <mailto:j.schoenwaelder@jacobs-university.de>

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

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

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

       Copyright (c) 2014 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 7407; see
       the RFC itself for full legal notices.";

    reference
      "RFC 3415: View-based Access Control Model (VACM) for the
         Simple Network Management Protocol (SNMP)";

    revision 2014-12-10 {
      description
        "Initial revision.";
      reference
        "RFC 7407: A YANG Data Model for SNMP Configuration";
    }

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

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

```
      }

      augment /snmp:snmp {

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

          list group {
            key name;
            description
              "VACM groups.

               This data model has a different structure than the MIB.
               Groups are explicitly defined in this list, and group
               members are defined in the 'member' list (mapped to
               vacmSecurityToGroupTable), and access for the group is
               defined in the 'access' list (mapped to
               vacmAccessTable).";
            reference
              "RFC 3415: View-based Access Control Model (VACM) for the
                  Simple Network Management Protocol (SNMP).
                  SNMP-VIEW-BASED-ACM-MIB.vacmSecurityToGroupTable
                  SNMP-VIEW-BASED-ACM-MIB.vacmAccessTable";

            leaf name {
              type group-name;
              description
                "The name of this VACM group.";
              reference
                "RFC 3415: View-based Access Control Model (VACM) for the
                    Simple Network Management Protocol (SNMP).
                    SNMP-VIEW-BASED-ACM-MIB.vacmGroupName";
            }

            list member {
              key "security-name";
              description
                "A member of this VACM group.

                 A specific combination of security-name and
                 security-model MUST NOT be present in more than
                 one group.";
              reference
                "RFC 3415: View-based Access Control Model (VACM) for the
                    Simple Network Management Protocol (SNMP).
                    SNMP-VIEW-BASED-ACM-MIB.vacmSecurityToGroupTable";
```

```
              leaf security-name {
                type snmp:security-name;
                description
                  "The securityName of a group member.";
                reference
                  "RFC 3415: View-based Access Control Model (VACM) for
                      the Simple Network Management Protocol (SNMP).
                      SNMP-VIEW-BASED-ACM-MIB.vacmSecurityName";
              }

              leaf-list security-model {
                type snmp:security-model;
                min-elements 1;
                description
                  "The security models under which this security-name
                   is a member of this group.";
                reference
                  "RFC 3415: View-based Access Control Model (VACM) for
                      the Simple Network Management Protocol (SNMP).
                      SNMP-VIEW-BASED-ACM-MIB.vacmSecurityModel";
              }
            }

            list access {
              key "context security-model security-level";
              description
                "Definition of access right for groups.";
              reference
                "RFC 3415: View-based Access Control Model (VACM) for
                    the Simple Network Management Protocol (SNMP).
                    SNMP-VIEW-BASED-ACM-MIB.vacmAccessTable";

              leaf context {
                type snmp:context-name;
                description
                  "The context (prefix) under which the access rights
                   apply.";
                reference
                  "RFC 3415: View-based Access Control Model (VACM) for
                      the Simple Network Management Protocol (SNMP).
                      SNMP-VIEW-BASED-ACM-MIB.vacmAccessContextPrefix";
              }

              leaf context-match {
                type enumeration {
                  enum exact { value 1; }
                  enum prefix { value 2; }
                }
```

```
            default exact;
            reference
              "RFC 3415: View-based Access Control Model (VACM) for
                 the Simple Network Management Protocol (SNMP).
                 SNMP-VIEW-BASED-ACM-MIB.vacmAccessContextMatch";
          }

          leaf security-model {
            type snmp:security-model-or-any;
            description
              "The security model under which the access rights
              apply.";
            reference
              "RFC 3415: View-based Access Control Model (VACM) for
                 the Simple Network Management Protocol (SNMP).
                 SNMP-VIEW-BASED-ACM-MIB.vacmAccessSecurityModel";
          }

          leaf security-level {
            type snmp:security-level;
            description
              "The minimum security level under which the access
               rights apply.";
            reference
              "RFC 3415: View-based Access Control Model (VACM) for
                 the Simple Network Management Protocol (SNMP).
                 SNMP-VIEW-BASED-ACM-MIB.vacmAccessSecurityLevel";
          }

          leaf read-view {
            type view-name;
            description
              "The name of the MIB view of the SNMP context
              authorizing read access.  If this leaf does not
              exist in a configuration, it maps to a zero-length
              vacmAccessReadViewName.

              Implementations MAY restrict the values of this
              leaf to be one of the available values of
              /snmp/vacm/view/name in a valid configuration.";
            reference
              "RFC 3415: View-based Access Control Model (VACM) for
                 the Simple Network Management Protocol (SNMP).
                 SNMP-VIEW-BASED-ACM-MIB.vacmAccessReadViewName";
          }

          leaf write-view {
            type view-name;
```

```
            description
              "The name of the MIB view of the SNMP context
               authorizing write access.  If this leaf does not
               exist in a configuration, it maps to a zero-length
               vacmAccessWriteViewName.

               Implementations MAY restrict the values of this
               leaf to be one of the available values of
               /snmp/vacm/view/name in a valid configuration.";
            reference
              "RFC 3415: View-based Access Control Model (VACM) for
                  the Simple Network Management Protocol (SNMP).
                  SNMP-VIEW-BASED-ACM-MIB.vacmAccessWriteViewName";
          }

          leaf notify-view {
            type view-name;
            description
              "The name of the MIB view of the SNMP context
               authorizing notify access.  If this leaf does not
               exist in a configuration, it maps to a zero-length
               vacmAccessNotifyViewName.

               Implementations MAY restrict the values of this
               leaf to be one of the available values of
               /snmp/vacm/view/name in a valid configuration.";
            reference
              "RFC 3415: View-based Access Control Model (VACM) for
                      the Simple Network Management Protocol (SNMP).
                      SNMP-VIEW-BASED-ACM-MIB.vacmAccessNotifyViewName";
          }
        }
      }

      list view {
        key name;
        description
          "Definition of MIB views.";
        reference
          "RFC 3415: View-based Access Control Model (VACM) for
              the Simple Network Management Protocol (SNMP).
              SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyTable";

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

```
            reference
              "RFC 3415: View-based Access Control Model (VACM) for
                 the Simple Network Management Protocol (SNMP).
                 SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyName";
          }

          leaf-list include {
            type snmp:wildcard-object-identifier;
            description
              "A family of subtrees included in this MIB view.";
            reference
              "RFC 3415: View-based Access Control Model (VACM) for
                 the Simple Network Management Protocol (SNMP).
                 SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilySubtree
                 SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyMask
                 SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyType";
          }

          leaf-list exclude {
            type snmp:wildcard-object-identifier;
            description
              "A family of subtrees excluded from this MIB view.";
            reference
              "RFC 3415: View-based Access Control Model (VACM) for
                 the Simple Network Management Protocol (SNMP).
                 SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilySubtree
                 SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyMask
                 SNMP-VIEW-BASED-ACM-MIB.vacmViewTreeFamilyType";
          }
        }
      }
    }
  }

  <CODE ENDS>
```

4.10. Submodule 'ietf-snmp-usm'

   This YANG submodule imports YANG extensions from [RFC6536].

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

```
   submodule ietf-snmp-usm {

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

```
import ietf-yang-types {
  prefix yang;
}
import ietf-netconf-acm {
  prefix nacm;
}

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

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

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

   WG Chair: Thomas Nadeau
             <mailto:tnadeau@lucidvision.com>

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

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

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

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

   Copyright (c) 2014 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 7407; see
   the RFC itself for full legal notices.";
```

```
     reference
       "RFC 3414: User-based Security Model (USM) for version 3 of the
          Simple Network Management Protocol (SNMPv3)";

     revision 2014-12-10 {
       description
         "Initial revision.";
       reference
         "RFC 7407: A YANG Data Model for SNMP Configuration";
     }

     grouping key {
       leaf key {
         type yang:hex-string;
         mandatory true;
         nacm:default-deny-all;
         description
           "Localized key specified as a list of colon-specified
            hexadecimal octets.";
       }
     }

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

         reference
           "RFC 3414: User-based Security Model (USM) for version 3
               of the Simple Network Management Protocol (SNMPv3).
               SNMP-USER-BASED-SM-MIB.usmUserTable";

         leaf name {
           type snmp:identifier;
           reference
             "RFC 3414: User-based Security Model (USM) for version 3
                 of the Simple Network Management Protocol (SNMPv3).
                 SNMP-USER-BASED-SM-MIB.usmUserName";
         }
         container auth {
           presence "enables authentication";
           description
             "Enables authentication of the user.";
           choice protocol {
             mandatory true;
             reference
               "RFC 3414: User-based Security Model (USM) for version 3
                   of the Simple Network Management Protocol (SNMPv3).
                   SNMP-USER-BASED-SM-MIB.usmUserAuthProtocol";
```

```
            container md5 {
              uses key;
              reference
                "RFC 3414: User-based Security Model (USM) for
                    version 3 of the Simple Network Management Protocol
                    (SNMPv3).
                    SNMP-USER-BASED-SM-MIB.usmHMACMD5AuthProtocol";
            }
            container sha {
              uses key;
              reference
                "RFC 3414: User-based Security Model (USM) for
                    version 3 of the Simple Network Management Protocol
                    (SNMPv3).
                    SNMP-USER-BASED-SM-MIB.usmHMACSHAAuthProtocol";
            }
          }
        }
        container priv {
          must "../auth" {
            error-message
              "when privacy (confidentiality) is used, "
            + "authentication must also be used";
          }
          presence "enables encryption";
          description
            "Enables encryption of SNMP messages.";

          choice protocol {
            mandatory true;
            reference
              "RFC 3414: User-based Security Model (USM) for version 3
                  of the Simple Network Management Protocol (SNMPv3).
                  SNMP-USER-BASED-SM-MIB.usmUserPrivProtocol";
            container des {
              uses key;
              reference
                "RFC 3414: User-based Security Model (USM) for
                    version 3 of the Simple Network Management Protocol
                    (SNMPv3).
                    SNMP-USER-BASED-SM-MIB.usmDESPrivProtocol";
            }
            container aes {
              uses key;
```

```
               reference
                 "RFC 3826: The Advanced Encryption Standard (AES)
                     Cipher Algorithm in the SNMP User-based Security
                     Model.
                     SNMP-USM-AES-MIB.usmAesCfb128Protocol";
             }
           }
         }
       }
     }

     augment /snmp:snmp {

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

         list remote {
           key "engine-id";

           leaf engine-id {
             type snmp:engine-id;
             reference
               "RFC 3414: User-based Security Model (USM) for version 3
                   of the Simple Network Management Protocol (SNMPv3).
                   SNMP-USER-BASED-SM-MIB.usmUserEngineID";
           }

           uses user-list;
         }
       }
     }

     grouping usm-target-params {
       container usm {
         description
           "User-based SNMPv3 parameters type.

            Represents snmpTargetParamsMPModel '3' and
            snmpTargetParamsSecurityModel '3'.";
         leaf user-name {
           type snmp:security-name;
           mandatory true;
```

```
                 reference
                   "RFC 3413: Simple Network Management Protocol (SNMP).
                      Applications.
                      SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
             }
             leaf security-level {
               type snmp:security-level;
               mandatory true;
               reference
                   "RFC 3413: Simple Network Management Protocol (SNMP).
                      Applications.
                      SNMP-TARGET-MIB.snmpTargetParamsSecurityLevel";
             }
           }
         }

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

     }

     <CODE ENDS>

4.11.  Submodule 'ietf-snmp-tsm'

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

     submodule ietf-snmp-tsm {

       belongs-to ietf-snmp {
         prefix snmp;
       }

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

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

       contact
         "WG Web:   <http://tools.ietf.org/wg/netmod/>
          WG List:  <mailto:netmod@ietf.org>
```

```
        WG Chair: Thomas Nadeau
                  <mailto:tnadeau@lucidvision.com>

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

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

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

   description
     "This submodule contains a collection of YANG definitions for
      configuring the Transport Security Model (TSM) of SNMP.

      Copyright (c) 2014 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 7407; see
      the RFC itself for full legal notices.";

   reference
     "RFC 5591: Transport Security Model for the
         Simple Network Management Protocol (SNMP)";

   revision 2014-12-10 {
     description
       "Initial revision.";
     reference
       "RFC 7407: A YANG Data Model for SNMP Configuration";
   }

   feature tsm {
     description
       "A server implements this feature if it supports the
        Transport Security Model for SNMP.";
     reference
       "RFC 5591: Transport Security Model for the
           Simple Network Management Protocol (SNMP)";
   }
```

```
      augment /snmp:snmp {
        if-feature tsm;
        container tsm {
          description
            "Configuration of the Transport Security Model.";

          leaf use-prefix {
            type boolean;
            default false;
            reference
              "RFC 5591: Transport Security Model for the Simple
                  Network Management Protocol (SNMP).
                  SNMP-TSM-MIB.snmpTsmConfigurationUsePrefix";
          }
        }
      }

      grouping tsm-target-params {
        container tsm {
          description
            "Transport-based security SNMPv3 parameters type.

             Represents snmpTargetParamsMPModel '3' and
             snmpTargetParamsSecurityModel '4'.";
          leaf security-name {
            type snmp:security-name;
            mandatory true;
            reference
              "RFC 3413: Simple Network Management Protocol (SNMP).
                  Applications.
                  SNMP-TARGET-MIB.snmpTargetParamsSecurityName";
          }
          leaf security-level {
            type snmp:security-level;
            mandatory true;
            reference
              "RFC 3413: Simple Network Management Protocol (SNMP).
                  Applications.
                  SNMP-TARGET-MIB.snmpTargetParamsSecurityLevel";
          }
        }
      }

      augment /snmp:snmp/snmp:target-params/snmp:params {
        if-feature tsm;
        case tsm {
          uses tsm-target-params;
        }
```

```
          }

      }

      <CODE ENDS>

4.12.  Submodule 'ietf-snmp-tls'

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

      submodule ietf-snmp-tls {

        belongs-to ietf-snmp {
          prefix snmp;
        }

        import ietf-inet-types {
          prefix inet;
        }
        import ietf-x509-cert-to-name {
          prefix x509c2n;
        }

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

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

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

           WG Chair: Thomas Nadeau
                     <mailto:tnadeau@lucidvision.com>

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

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

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

         description
           "This submodule contains a collection of YANG definitions for
            configuring the Transport Layer Security Transport Model (TLSTM)
            of SNMP.

            Copyright (c) 2014 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 7407; see
            the RFC itself for full legal notices.";

         reference
           "RFC 6353: Transport Layer Security (TLS) Transport Model for
               the Simple Network Management Protocol (SNMP)";

         revision 2014-12-10 {
           description
             "Initial revision.";
           reference
             "RFC 7407: A YANG Data Model for SNMP Configuration";
         }

         feature tlstm {
           description
             "A server implements this feature if it supports the
             Transport Layer Security Transport Model for SNMP.";
           reference
             "RFC 6353: Transport Layer Security (TLS) Transport Model for
                 the Simple Network Management Protocol (SNMP)";
         }

         augment /snmp:snmp/snmp:engine/snmp:listen/snmp:transport {
           if-feature tlstm;
           case tls {
             container tls {
               description
                 "A list of IPv4 and IPv6 addresses and ports to which the
                  engine listens for SNMP messages over TLS.";

```
            leaf ip {
              type inet:ip-address;
              mandatory true;
              description
                "The IPv4 or IPv6 address on which the engine listens
                 for SNMP messages over TLS.";
            }
            leaf port {
              type inet:port-number;
              description
                "The TCP port on which the engine listens for SNMP
                 messages over TLS.

                 If the port is not configured, an engine that
                 acts as a Command Responder uses port 10161, and
                 an engine that acts as a Notification Receiver
                 uses port 10162.";
            }
          }
        }
        case dtls {
          container dtls {
            description
              "A list of IPv4 and IPv6 addresses and ports to which the
               engine listens for SNMP messages over DTLS.";

            leaf ip {
              type inet:ip-address;
              mandatory true;
              description
                "The IPv4 or IPv6 address on which the engine listens
                 for SNMP messages over DTLS.";
            }
            leaf port {
              type inet:port-number;
              description
                "The UDP port on which the engine listens for SNMP
                 messages over DTLS.

                 If the port is not configured, an engine that
                 acts as a Command Responder uses port 10161, and
                 an engine that acts as a Notification Receiver
                 uses port 10162.";
            }
          }
        }
      }
```

```
     augment /snmp:snmp {
       if-feature tlstm;
       container tlstm {
         uses x509c2n:cert-to-name {
           description
             "Defines how certificates are mapped to names.  The
              resulting name is used as a security name.";
           refine cert-to-name/map-type {
             description
               "Mappings that use the snmpTlstmCertToTSNData column
                need to augment the cert-to-name list with
                additional configuration objects corresponding
                to the snmpTlstmCertToTSNData value.  Such objects
                should use the 'when' statement to make them
                conditional based on the map-type.";
           }
         }
       }
     }

     grouping tls-transport {
       leaf ip {
         type inet:host;
         mandatory true;
         reference
           "RFC 3413: Simple Network Management Protocol (SNMP).
              Applications.
              SNMP-TARGET-MIB.snmpTargetAddrTAddress
            RFC 6353: Transport Layer Security (TLS) Transport Model
              for the Simple Network Management Protocol (SNMP).
              SNMP-TLS-TM-MIB.SnmpTLSAddress";
       }
       leaf port {
         type inet:port-number;
         default 10161;
         reference
           "RFC 3413: Simple Network Management Protocol (SNMP).
              Applications.
              SNMP-TARGET-MIB.snmpTargetAddrTAddress
            RFC 6353: Transport Layer Security (TLS) Transport Model
              for the Simple Network Management Protocol (SNMP).
              SNMP-TLS-TM-MIB.SnmpTLSAddress";
       }
       leaf client-fingerprint {
         type x509c2n:tls-fingerprint;
         reference
           "RFC 6353: Transport Layer Security (TLS) Transport Model
              for the Simple Network Management Protocol (SNMP).
```

```
                     SNMP-TLS-TM-MIB.snmpTlstmParamsClientFingerprint";
          }
          leaf server-fingerprint {
            type x509c2n:tls-fingerprint;
            reference
              "RFC 6353: Transport Layer Security (TLS) Transport Model
                  for the Simple Network Management Protocol (SNMP).
                  SNMP-TLS-TM-MIB.snmpTlstmAddrServerFingerprint";
          }
          leaf server-identity {
            type snmp:admin-string;
            reference
              "RFC 6353: Transport Layer Security (TLS) Transport Model
                  for the Simple Network Management Protocol (SNMP).
                  SNMP-TLS-TM-MIB.snmpTlstmAddrServerIdentity";
          }
       }

       augment /snmp:snmp/snmp:target/snmp:transport {
         if-feature tlstm;
         case tls {
           reference
             "RFC 6353: Transport Layer Security (TLS) Transport Model
                 for the Simple Network Management Protocol (SNMP).
                 SNMP-TLS-TM-MIB.snmpTLSTCPDomain";
           container tls {
             uses tls-transport;
           }
         }
       }

       augment /snmp:snmp/snmp:target/snmp:transport {
         if-feature tlstm;
         case dtls {
           reference
             "RFC 6353: Transport Layer Security (TLS) Transport Model
                 for the Simple Network Management Protocol (SNMP).
                 SNMP-TLS-TM-MIB.snmpDTLSUDPDomain";
           container dtls {
             uses tls-transport;
           }
         }
       }
     }

     <CODE ENDS>
```

4.13. Submodule 'ietf-snmp-ssh'

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

   submodule ietf-snmp-ssh {

     belongs-to ietf-snmp {
       prefix snmp;
     }

     import ietf-inet-types {
       prefix inet;
     }

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

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

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

        WG Chair: Thomas Nadeau
                  <mailto:tnadeau@lucidvision.com>

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

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

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

   description
     "This submodule contains a collection of YANG definitions for
      configuring the Secure Shell Transport Model (SSHTM)
      of SNMP.

      Copyright (c) 2014 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 7407; see
      the RFC itself for full legal notices.";

  reference
    "RFC 5592: Secure Shell Transport Model for the
       Simple Network Management Protocol (SNMP)";

  revision 2014-12-10 {
    description
      "Initial revision.";
    reference
      "RFC 7407: A YANG Data Model for SNMP Configuration";
  }

  feature sshtm {
    description
      "A server implements this feature if it supports the
      Secure Shell Transport Model for SNMP.";
    reference
      "RFC 5592: Secure Shell Transport Model for the
         Simple Network Management Protocol (SNMP)";
  }

  augment /snmp:snmp/snmp:engine/snmp:listen/snmp:transport {
    if-feature sshtm;
    case ssh {
      container ssh {
        description
          "The IPv4 or IPv6 address and port to which the
           engine listens for SNMP messages over SSH.";

        leaf ip {
          type inet:ip-address;
          mandatory true;
          description
            "The IPv4 or IPv6 address on which the engine listens
             for SNMP messages over SSH.";
        }
        leaf port {
          type inet:port-number;
          description
            "The TCP port on which the engine listens for SNMP
             messages over SSH.
```

```
                     If the port is not configured, an engine that
                     acts as a Command Responder uses port 5161, and
                     an engine that acts as a Notification Receiver
                     uses port 5162.";
              }
            }
          }
        }

      augment /snmp:snmp/snmp:target/snmp:transport {
        if-feature sshtm;
        case ssh {
          reference
            "RFC 5592: Secure Shell Transport Model for the
               Simple Network Management Protocol (SNMP).
               SNMP-SSH-TM-MIB.snmpSSHDomain";
          container ssh {
            leaf ip {
              type inet:host;
              mandatory true;
              reference
                "RFC 3413: Simple Network Management Protocol (SNMP).
                   Applications.
                   SNMP-TARGET-MIB.snmpTargetAddrTAddress
                 RFC 5592: Secure Shell Transport Model for the
                   Simple Network Management Protocol (SNMP).
                   SNMP-SSH-TM-MIB.SnmpSSHAddress";
            }
            leaf port {
              type inet:port-number;
              default 5161;
              reference
                "RFC 3413: Simple Network Management Protocol (SNMP).
                   Applications.
                   SNMP-TARGET-MIB.snmpTargetAddrTAddress
                 RFC 5592: Secure Shell Transport Model for the
                   Simple Network Management Protocol (SNMP).
                   SNMP-SSH-TM-MIB.SnmpSSHAddress";
            }
            leaf username {
              type string;
              reference
                "RFC 3413: Simple Network Management Protocol (SNMP).
                   Applications.
                   SNMP-TARGET-MIB.snmpTargetAddrTAddress
                 RFC 5592: Secure Shell Transport Model for the
                   Simple Network Management Protocol (SNMP).
                   SNMP-SSH-TM-MIB.SnmpSSHAddress";
```

```
                }
              }
            }
          }
        }
```


    <CODE ENDS>

5. IANA Considerations

    This document registers two URIs in the "IETF XML Registry"
    [RFC3688]. Following the format in RFC 3688, the following
    registrations have been made.

        URI: urn:ietf:params:xml:ns:yang:ietf-snmp
        Registrant Contact: The NETMOD WG of the IETF.
        XML: N/A, the requested URI is an XML namespace.

        URI: urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name
        Registrant Contact: The NETMOD WG of the IETF.
        XML: N/A, the requested URI is an XML namespace.

    This document registers the following YANG modules in the "YANG
    Module Names" registry [RFC6020].

        name: ietf-snmp
        namespace: urn:ietf:params:xml:ns:yang:ietf-snmp
        prefix: snmp
        reference: RFC 7407

        name: ietf-x509-cert-to-name
        namespace: urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name
        prefix: x509c2n
        reference: RFC 7407
    The document registers the following YANG submodules in the "YANG
    Module Names" registry [RFC6020].

        name: ietf-snmp-common
        parent: ietf-snmp
        reference: RFC 7407

        name: ietf-snmp-engine
        parent: ietf-snmp
        reference: RFC 7407

```
      name:         ietf-snmp-community
      parent:       ietf-snmp
      reference:    RFC 7407

      name:         ietf-snmp-notification
      parent:       ietf-snmp
      reference:    RFC 7407

      name:         ietf-snmp-target
      parent:       ietf-snmp
      reference:    RFC 7407

      name:         ietf-snmp-vacm
      parent:       ietf-snmp
      reference:    RFC 7407

      name:         ietf-snmp-usm
      parent:       ietf-snmp
      reference:    RFC 7407

      name:         ietf-snmp-tsm
      parent:       ietf-snmp
      reference:    RFC 7407

      name:         ietf-snmp-tls
      parent:       ietf-snmp
      reference:    RFC 7407

      name:         ietf-snmp-ssh
      parent:       ietf-snmp
      reference:    RFC 7407
```

6. Security Considerations

   The YANG module and submodules defined in this memo are designed to
   be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF
   layer is the secure transport layer and the mandatory to implement
   secure transport is SSH [RFC6242]. The NETCONF access control model
   [RFC6536] provides the means to restrict access for particular
   NETCONF users to a pre-configured subset of all available NETCONF
   protocol operations and content.

   There are a number of data nodes defined in the YANG module and
   submodules which are writable/creatable/deletable (i.e., config true,
   which is the default). These data nodes may be considered sensitive
   or vulnerable in some network environments. Write operations (e.g.,

   edit-config) to these data nodes without proper protection can have a
   negative effect on network operations. These are the subtrees and
   data nodes and their sensitivity/vulnerability:

   o The "/snmp/engine" subtree contains the configuration of general
      parameters of an SNMP engine such as the endpoints to listen on,
      the transports and SNMP versions enabled, or the engine's
      identity. Write access to this subtree should only be granted to
      entities configuring general SNMP engine parameters.

   o The "/snmp/target" subtree contains the configuration of SNMP
      targets and, in particular, which transports to use and their
      security parameters. Write access to this subtree should only be
      granted to the security administrator and entities configuring
      SNMP notification forwarding behavior.

   o The "/snmp/notify" and "/snmp/notify-filter-profile" subtrees
      contain the configuration for the SNMP notification forwarding and
      filtering mechanism. Write access to these subtrees should only
      be granted to entities configuring SNMP notification forwarding
      behavior.

   o The "/snmp/proxy" subtree contains the configuration for SNMP
      proxies. Write access to this subtree should only be granted to
      entities configuring SNMP proxies.

   o The "/snmp/community" subtree contains the configuration of the
      Community-based Security Model. Write access to this subtree
      should only be granted to the security administrator.

   o The "/snmp/usm" subtree contains the configuration of the User-
      based Security Model. Write access to this subtree should only be
      granted to the security administrator.

   o The "/snmp/tsm" subtree contains the configuration of the
      Transport Layer Security (TLS) Transport Model for SNMP. Write
      access to this subtree should only be granted to the security
      administrator.

   o The "/snmp/tlstm" subtree contains the configuration of the SNMP
      transport over (D)TLS and, in particular, the configuration of how
      certificates are mapped to SNMP security names. Write access to
      this subtree should only be granted to the security administrator.

   o The "/snmp/vacm" subtree contains the configuration of the View-
      based Access Control Model used by SNMP to authorize access to
      management information via SNMP. Write access to this subtree
      should only be granted to the security administrator.

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

o The "/snmp/engine" subtree exposes general information about an
   SNMP engine such as which version(s) of SNMP are enabled or which
   transports are enabled.

o The "/snmp/target" subtree exposes information about which
   transports are used to reach certain SNMP targets and which
   transport-specific parameters are used.

o The "/snmp/notify" and "/snmp/notify-filter-profile" subtrees
   expose information about how notifications are filtered and
   forwarded to notification targets.

o The "/snmp/proxy" subtree exposes information about proxy
   relationships.

o The "/snmp/community", "/snmp/usm", "/snmp/tsm", "/snmp/tlstm",
   and "/snmp/vacm" subtrees are specifically sensitive since they
   expose information about the authentication and authorization
   policy used by an SNMP engine.

Changes to the SNMP access control rules should be done in an atomic
way (through a single edit-config or a single commit), or care must
be taken that they are done in a sequence that does not temporarily
open access to resources. Implementations supporting SNMP write
access must ensure that any SNMP access control rule changes over
NETCONF are also atomic to the SNMP instrumentation. In particular,
changes involving an internal delete/create cycle (e.g., to move a
user to a different group) must be done with sufficient protections
such that even a power fail immediately after the delete does not
leave the administrator locked out.

Security administrators need to ensure that NETCONF access control
rules and SNMP access control rules implement a consistent security
policy. Specifically, the SNMP access control rules should prevent
accidental leakage of sensitive security parameters such as community
strings. See the Security Considerations section of [RFC3584] for
further details.

7. References

7.1. Normative References

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119, March 1997,
               <http://www.rfc-editor.org/info/rfc2119>.

   [RFC6020]   Bjorklund, M., "YANG - A Data Modeling Language for the
               Network Configuration Protocol (NETCONF)", RFC 6020,
               October 2010, <http://www.rfc-editor.org/info/rfc6020>.

   [RFC6241]   Enns, R., Bjorklund, M., Schoenwaelder, J., and A.
               Bierman, "Network Configuration Protocol (NETCONF)", RFC
               6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

   [RFC6242]   Wasserman, M., "Using the NETCONF Protocol over Secure
               Shell (SSH)", RFC 6242, June 2011,
               <http://www.rfc-editor.org/info/rfc6242>.

   [RFC6536]   Bierman, A. and M. Bjorklund, "Network Configuration
               Protocol (NETCONF) Access Control Model", RFC 6536, March
               2012, <http://www.rfc-editor.org/info/rfc6536>.

   [RFC6991]   Schoenwaelder, J., "Common YANG Data Types", RFC 6991,
               July 2013, <http://www.rfc-editor.org/info/rfc6991>.

7.2. Informative References

   [RFC3411]   Harrington, D., Presuhn, R., and B. Wijnen, "An
               Architecture for Describing Simple Network Management
               Protocol (SNMP) Management Frameworks", STD 62, RFC 3411,
               December 2002, <http://www.rfc-editor.org/info/rfc3411>.

   [RFC3412]   Case, J., Harrington, D., Presuhn, R., and B. Wijnen,
               "Message Processing and Dispatching for the Simple Network
               Management Protocol (SNMP)", STD 62, RFC 3412, December
               2002, <http://www.rfc-editor.org/info/rfc3412>.

   [RFC3413]   Levi, D., Meyer, P., and B. Stewart, "Simple Network
               Management Protocol (SNMP) Applications", STD 62, RFC
               3413, December 2002,
               <http://www.rfc-editor.org/info/rfc3413>.

   [RFC3414]   Blumenthal, U. and B. Wijnen, "User-based Security Model
               (USM) for version 3 of the Simple Network Management
               Protocol (SNMPv3)", STD 62, RFC 3414, December 2002,
               <http://www.rfc-editor.org/info/rfc3414>.

   [RFC3415] Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based
              Access Control Model (VACM) for the Simple Network
              Management Protocol (SNMP)", STD 62, RFC 3415, December
              2002, <http://www.rfc-editor.org/info/rfc3415>.

   [RFC3417] Presuhn, R., "Transport Mappings for the Simple Network
              Management Protocol (SNMP)", STD 62, RFC 3417, December
              2002, <http://www.rfc-editor.org/info/rfc3417>.

   [RFC3418] Presuhn, R., "Management Information Base (MIB) for the
              Simple Network Management Protocol (SNMP)", STD 62, RFC
              3418, December 2002,
              <http://www.rfc-editor.org/info/rfc3418>.

   [RFC3419] Daniele, M. and J. Schoenwaelder, "Textual Conventions for
              Transport Addresses", RFC 3419, December 2002,
              <http://www.rfc-editor.org/info/rfc3419>.

   [RFC3584] Frye, R., Levi, D., Routhier, S., and B. Wijnen,
              "Coexistence between Version 1, Version 2, and Version 3
              of the Internet-standard Network Management Framework",
              BCP 74, RFC 3584, August 2003,
              <http://www.rfc-editor.org/info/rfc3584>.

   [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
              January 2004, <http://www.rfc-editor.org/info/rfc3688>.

   [RFC3826] Blumenthal, U., Maino, F., and K. McCloghrie, "The
              Advanced Encryption Standard (AES) Cipher Algorithm in the
              SNMP User-based Security Model", RFC 3826, June 2004,
              <http://www.rfc-editor.org/info/rfc3826>.

   [RFC5591] Harrington, D. and W. Hardaker, "Transport Security Model
              for the Simple Network Management Protocol (SNMP)", STD
              78, RFC 5591, June 2009,
              <http://www.rfc-editor.org/info/rfc5591>.

   [RFC5592] Harrington, D., Salowey, J., and W. Hardaker, "Secure
              Shell Transport Model for the Simple Network Management
              Protocol (SNMP)", RFC 5592, June 2009,
              <http://www.rfc-editor.org/info/rfc5592>.

   [RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport
              Model for the Simple Network Management Protocol (SNMP)",
              STD 78, RFC 6353, July 2011,
              <http://www.rfc-editor.org/info/rfc6353>.

   [RFC6643] Schoenwaelder, J., "Translation of Structure of Management
              Information Version 2 (SMIv2) MIB Modules to YANG
              Modules", RFC 6643, July 2012,
              <http://www.rfc-editor.org/info/rfc6643>.

Appendix A. Example Configurations

A.1. Engine Configuration Example

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

   <snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
     <engine>
       <enabled>true</enabled>
       <listen>
         <name>all-ipv4-udp</name>
         <udp>
           <ip>0.0.0.0</ip>
           <port>161</port>
         </udp>
       </listen>
       <listen>
         <name>all-ipv6-udp</name>
         <udp>
           <ip>::</ip>
           <port>161</port>
         </udp>
       </listen>
       <version>
         <v2c/>
         <v3/>
       </version>
       <engine-id>80:00:02:b8:04:61:62:63</engine-id>
     </engine>
   </snmp>

A.2. Community Configuration Example

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

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

```
      <name>management-station</name>
      <udp>
        <ip>2001:db8::abcd</ip>
        <port>161</port>
      </udp>
      <tag>blue</tag>
      <tag>community-public-access</tag>
      <target-params>v2c-public</target-params>
    </target>
    <target-params>
      <name>v2c-public</name>
      <v2c>
        <security-name>community-public</security-name>
      </v2c>
    </target-params>
  </snmp>
```

A.3. User-Based Security Model Configuration Example

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

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

```
          <user>
            <name>russ</name>
            <auth>
              <sha>
                <!--
                    The 'key' value is split into two lines to conform to
                    the RFC formatting rules.
                -->
                <key>66:95:fe:bc:92:88:e3:62:82:23:
                    5f:c7:15:1f:12:84:97:b3:8f:3f</key>
              </sha>
            </auth>
            <priv>
              <aes>
                <!--
                    The 'key' value is split into two lines to conform to
                    the RFC formatting rules.
                -->
                <key>66:95:fe:bc:92:88:e3:62:82:23:
                    5f:c7:15:1f:12:84</key>
              </aes>
            </priv>
          </user>
        </remote>
      </usm>
      <target>
        <name>bluebox</name>
        <udp>
          <ip>2001:db8::abcd</ip>
          <port>161</port>
        </udp>
        <tag>blue</tag>
        <target-params>matt-auth</target-params>
      </target>
      <target-params>
        <name>matt-auth</name>
        <usm>
          <user-name>matt</user-name>
          <security-level>auth-no-priv</security-level>
        </usm>
      </target-params>
    </snmp>
```

A.4. Target and Notification Configuration Example

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

   <snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
     <target>
       <name>addr1</name>
       <udp>
         <ip>192.0.2.3</ip>
         <port>162</port>
       </udp>
       <tag>group1</tag>
       <target-params>joe-auth</target-params>
     </target>
     <target>
       <name>addr2</name>
       <udp>
         <ip>192.0.2.6</ip>
         <port>162</port>
       </udp>
       <tag>group1</tag>
       <target-params>joe-auth</target-params>
     </target>
     <target>
       <name>addr3</name>
       <udp>
         <ip>192.0.2.9</ip>
         <port>162</port>
       </udp>
       <tag>group2</tag>
       <target-params>bob-priv</target-params>
     </target>
     <target-params>
       <name>joe-auth</name>
       <usm>
         <user-name>joe</user-name>
         <security-level>auth-no-priv</security-level>
       </usm>
     </target-params>
     <target-params>
       <name>bob-priv</name>
       <usm>
         <user-name>bob</user-name>
         <security-level>auth-priv</security-level>
       </usm>

```
      </target-params>
      <notify>
        <name>group1</name>
        <tag>group1</tag>
        <type>trap</type>
      </notify>
      <notify>
        <name>group2</name>
        <tag>group2</tag>
        <type>trap</type>
      </notify>
    </snmp>
```

A.5. Proxy Configuration Example

   Below is an XML instance document showing the configuration of a
   proxy forwarder application. It proxies SNMPv2c messages from
   command generators to a file server running an SNMPv1 agent that
   recognizes two community strings, "private" and "public", with
   different associated read views. The file server is represented as
   two "target" instances, one for each community string.

   If the proxy receives an SNMPv2c message with the community string
   "public" from a device in the "Office Network" or "Home Office
   Network", it gets tagged as "trusted", and the proxy uses the
   "private" community string when sending the message to the file
   server. Other SNMPv2c messages with the community string "public"
   get tagged as "non-trusted", and the proxy uses the "public"
   community string for these messages. There is also a special
   "backdoor" community string that can be used from any location to get
   "trusted" access.

   The "Office Network" and "Home Office Network" are represented as two
   "target" instances. These "target" instances have target-params
   "none", which refers to a non-existing target-params entry.

```
    <snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
      <target>
        <name>File Server (private)</name>
        <udp>
          <ip>192.0.2.1</ip>
        </udp>
        <target-params>v1-private</target-params>
      </target>
      <target>
        <name>File Server (public)</name>
        <udp>
          <ip>192.0.2.1</ip>
```

```
        </udp>
        <target-params>v1-public</target-params>
      </target>
      <target>
        <name>Office Network</name>
        <udp>
          <ip>192.0.2.0</ip>
          <prefix-length>24</prefix-length>
        </udp>
        <tag>office</tag>
        <target-params>none</target-params>
      </target>
      <target>
        <name>Home Office Network</name>
        <udp>
          <ip>203.0.113.0</ip>
          <prefix-length>24</prefix-length>
        </udp>
        <tag>home-office</tag>
        <target-params>none</target-params>
      </target>
      <target-params>
        <name>v1-private</name>
        <v1>
          <security-name>private</security-name>
        </v1>
      </target-params>
      <target-params>
        <name>v1-public</name>
        <v1>
          <security-name>public</security-name>
        </v1>
      </target-params>
      <target-params>
        <name>v2c-public</name>
        <v2c>
          <security-name>public</security-name>
        </v2c>
      </target-params>

      <!--
          Communities c1, c2, c3, and c4 are used for incoming messages
          that should be forwarded.

          Communities c3 and c5 are used for outgoing messages to the
          file server.
      -->
      <community>
```

```
      <index>c1</index>
      <security-name>public</security-name>
      <engine-id>80:00:61:81:c8</engine-id>
      <context>trusted</context>
      <target-tag>office</target-tag>
    </community>
    <community>
      <index>c2</index>
      <security-name>public</security-name>
      <engine-id>80:00:61:81:c8</engine-id>
      <context>trusted</context>
      <target-tag>home-office</target-tag>
    </community>
    <community>
      <index>c3</index>
      <security-name>public</security-name>
      <engine-id>80:00:61:81:c8</engine-id>
      <context>not-trusted</context>
    </community>
    <community>
      <index>c4</index>
      <text-name>backdoor</text-name>
      <security-name>public</security-name>
      <engine-id>80:00:61:81:c8</engine-id>
      <context>trusted</context>
    </community>
    <community>
      <index>c5</index>
      <security-name>private</security-name>
      <engine-id>80:00:61:81:c8</engine-id>
      <context>trusted</context>
    </community>

    <proxy>
      <name>p1</name>
      <type>read</type>
      <context-engine-id>80:00:61:81:c8</context-engine-id>
      <context-name>trusted</context-name>
      <target-params-in>v2c-public</target-params-in>
      <single-target-out>File Server (private)</single-target-out>
    </proxy>
    <proxy>
      <name>p2</name>
      <type>read</type>
      <context-engine-id>80:00:61:81:c8</context-engine-id>
      <context-name>not-trusted</context-name>
      <target-params-in>v2c-public</target-params-in>
      <single-target-out>File Server (public)</single-target-out>
```

```
      </proxy>
   </snmp>
```

   If an SNMPv2c Get request with community string "public" is received
   from an IP address tagged as "office" or "home-office", or if the
   request is received from anywhere else with community string
   "backdoor", the implied context is "trusted" so proxy entry "p1"
   matches. The request is forwarded to the file server as SNMPv1 with
   community "private" using community table entry "c5" for outbound
   params lookup.

   If an SNMPv2c Get request with community string "public" is received
   from any other IP address, the implied context is "not-trusted" so
   proxy entry "p2" matches, and the request is forwarded to the file
   server as SNMPv1 with community "public".

A.6. View-Based Access Control Model Configuration Example

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

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

```
          </view>
          <view>
            <name>restricted</name>
            <include>1.3.6.1</include>
          </view>
        </vacm>
      </snmp>

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

      <snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp">
        <vacm>
          <group>
            <name>initial</name>
            <member>
              <security-name>initial</security-name>
              <security-model>usm</security-model>
            </member>
            <access>
              <context></context>
              <security-model>usm</security-model>
              <security-level>no-auth-no-priv</security-level>
              <read-view>restricted</read-view>
              <notify-view>restricted</notify-view>
            </access>
            <access>
              <context></context>
              <security-model>usm</security-model>
              <security-level>auth-no-priv</security-level>
              <read-view>internet</read-view>
              <write-view>internet</write-view>
              <notify-view>internet</notify-view>
            </access>
          </group>
          <view>
            <name>initial</name>
            <include>1.3.6.1</include>
          </view>
          <view>
            <name>restricted</name>
            <include>1.3.6.1.2.1.1</include>
            <include>1.3.6.1.2.1.11</include>
            <include>1.3.6.1.6.3.10.2.1</include>
            <include>1.3.6.1.6.3.11.2.1</include>
            <include>1.3.6.1.6.3.15.1.1</include>
          </view>
        </vacm>
```

```
        </snmp>

A.7.  Transport Layer Security Transport Model Configuration Example

   Below is an XML instance document showing the configuration of the
   mapping of certificate to security name (see Appendices A.2 and A.3
   of [RFC6353]).

   <snmp xmlns="urn:ietf:params:xml:ns:yang:ietf-snmp"
         xmlns:x509c2n=
           "urn:ietf:params:xml:ns:yang:ietf-x509-cert-to-name">
     <tlstm>
       <cert-to-name>
         <id>1</id>
         <fingerprint>11:0A:05:11:00</fingerprint>
         <map-type>x509c2n:san-any</map-type>
       </cert-to-name>
       <cert-to-name>
         <id>2</id>
         <fingerprint>11:0A:05:11:00</fingerprint>
         <map-type>x509c2n:specified</map-type>
         <name>
           Joe Cool
         </name>
       </cert-to-name>
     </tlstm>
   </snmp>
```

Acknowledgments

Authors' Addresses

   Martin Bjorklund
   Tail-f Systems

   EMail: mbj@tail-f.com

   Juergen Schoenwaelder
   Jacobs University

   EMail: j.schoenwaelder@jacobs-university.de