Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 15, 2009

J. Schoenwaelder, Ed.
Jacobs University

July 14, 2008

Common YANG Data Types
draft-schoenw-netmod-yang-types-01

Status of this Memo

Copyright Notice

Abstract

This document introduces a collection of common data types to be used with the YANG data modeling language.

Table of Contents

1. Introduction

YANG [YANG] is a data modeling language used to model configuration and state data manipulated by the NETCONF [RFC4741] protocol. The YANG language supports a small set of built-in data types and provides mechanisms to derive other types from the built-in types.

This document introduces a collection of common data types derived from the built-in YANG data types. The definitions are organized in several YANG modules. The "yang-types" module contains generally useful data types. The "inet-types" module contains definitions that are relevant for the Internet protocol suite while the "ieee-types" module contains definitions that are relevant for IEEE 802 protocols.

Their derived types are generally designed to be applicable for modeling all areas of management information.

2. Key Words

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

3. Core YANG Derived Types

```
module yang-types {

    // XXX namespace to be allocated by IANA

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

    organization
        "YANG Language Design Team";

    contact
        "Juergen Schoenwaelder (Editor)
         <j.schoenwaelder@jacobs-university.de>";

    description
        "This module contains standard derived YANG types.";

    revision 2009-05-22 {
        description "Initial revision.";
    }

    /*
     * collection of counter and gauge types
     */

    typedef counter32 {
        type uint32;
        description
           "The counter32 type represents a non-negative integer
            which monotonically increases until it reaches a
            maximum value of 2^32-1 (4294967295 decimal), when it
            wraps around and starts increasing again from zero.

            Counters have no defined `initial' value, and thus, a
            single value of a counter has (in general) no information
            content.  Discontinuities in the monotonically increasing
            value normally occur at re-initialization of the
            management system, and at other times as specified in the
            description of an object instance using this type.  If
            such other times can occur, for example, the creation of
            an object instance of type counter32 at times other than
            re-initialization, then a corresponding object should be
            defined, with an appropriate type, to indicate the last
            discontinuity.

            The counter32 type should not be used for configuration
```

```
            objects. A default statement should not be used for
            attributes with a type value of counter32.";
        reference
           "RFC 2578 (STD 58)";
    }

    typedef zero-based-counter32 {
        type yang:counter32;
        default "0";
        description
            "The zero-based-counter32 type represents a counter32
             which has the defined `initial' value zero.";
        reference
           "RFC 2021";
    }

    typedef counter64 {
        type uint64;
        description
           "The counter64 type represents a non-negative integer
            which monotonically increases until it reaches a
            maximum value of 2^64-1 (18446744073709551615), when
            it wraps around and starts increasing again from zero.

            Counters have no defined `initial' value, and thus, a
            single value of a counter has (in general) no information
            content.  Discontinuities in the monotonically increasing
            value normally occur at re-initialization of the
            management system, and at other times as specified in the
            description of an object instance using this type.  If
            such other times can occur, for example, the creation of
            an object instance of type counter64 at times other than
            re-initialization, then a corresponding object should be
            defined, with an appropriate type, to indicate the last
            discontinuity.

            The counter64 type should not be used for configuration
            objects. A default statement should not be used for
            attributes with a type value of counter64.";
        reference
           "RFC 2578 (STD 58)";
    }

    typedef zero-based-counter64 {
        type yang:counter64;
        default "0";
        description
            "The zero-based-counter64 type represents a counter64
```

```
             which has the defined `initial' value zero.";
        reference
           "RFC 2856";
    }

    typedef gauge32 {
        type uint32;
        description
           "The gauge32 type represents a non-negative integer,
            which may increase or decrease, but shall never
            exceed a maximum value, nor fall below a minimum
            value.  The maximum value can not be greater than
            2^32-1 (4294967295 decimal), and the minimum value
            can not be smaller than 0.  The value of a gauge32
            has its maximum value whenever the information
            being modeled is greater than or equal to its
            maximum value, and has its minimum value whenever
            the information being modeled is smaller than or
            equal to its minimum value.  If the information
            being modeled subsequently decreases below
            (increases above) the maximum (minimum) value, the
            gauge32 also decreases (increases).";
        reference
           "RFC 2578 (STD 58)";
    }

    typedef gauge64 {
        type uint64;
        description
           "The gauge64 type represents a non-negative integer,
            which may increase or decrease, but shall never
            exceed a maximum value, nor fall below a minimum
            value.  The maximum value can not be greater than
            2^64-1 (18446744073709551615), and the minimum value
            can not be smaller than 0.  The value of a gauge64
            has its maximum value whenever the information
            being modeled is greater than or equal to its
            maximum value, and has its minimum value whenever
            the information being modeled is smaller than or
            equal to its minimum value.  If the information
            being modeled subsequently decreases below
            (increases above) the maximum (minimum) value, the
            gauge64 also decreases (increases).";
        reference
           "RFC 2856";
    }

    /*
```

```
    * collection of identifier related types
    */

   typedef uri {
       type string;
       description
         "A uri type represents Uniform Resource Identifier (URI)
          as defined by STD 66.

          Objects using this type MUST be in US-ASCII encoding, and
          MUST be normalized as described by RFC 3986 Sections
          6.2.1, 6.2.2.1, and 6.2.2.2.  All unnecessary
          percent-encoding is removed, and all case-insensitive
          characters are set to lowercase except for hexadecimal
          digits, which are normalized to uppercase as described in
          Section 6.2.2.1.

          The purpose of this normalization is to help provide unique
          URIs.  Note that this normalization is not sufficient to
          provide uniqueness.  Two URIs that are textually distinct
          after this normalization may still be equivalent.

          Objects using this type MAY restrict the schemes that they
          permit.  For example, 'data:' and 'urn:' schemes might not
          be appropriate.

          A zero-length URI is not a valid URI.  This can be used to
          express 'URI absent' where required, for example when used
          as an index field.";
       reference
         "RFC 3986 (STD 66), RFC 3305, and RFC 5017";
   }

   typedef object-identifier {
       type string {
           pattern '([0-1](\.[1-3]?[0-9]))|(2.(0|([1-9]\d*)))'
                 + '(\.(0|([1-9]\d*)))*';
       }
       description
         "The object-identifier type represents administratively
          assigned names in a registration-hierarchical-name tree.

          Values of this type are denoted as a sequence of numerical
          non-negative sub-identifier values. Each sub-identifier
          value MUST NOT exceed 2^32-1 (4294967295). Sub-identifiers
          are separated by single dots and without any intermediate
          white space.
```

```
            Although the number of sub-identifiers is not limited,
            module designers should realize that there may be
            implementations that stick with the SMIv1/v2 limit of 128
            sub-identifiers.";
        reference
           "ITU-T Recommendation X.660 / ISO/IEC 9834-1";
    }

    /*
     * collection of date and time related types
     */

    typedef date-and-time {
        type string {
            pattern '\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.d*)?'
                  + '(Z|(\+|-)\d{2}:\d{2})';
        }
        description
           'The date-and-time type is a profile of the ISO 8601
            standard for representation of dates and times using the
            Gregorian calendar. The format is most easily described
            using the following ABFN (see RFC 3339):

            date-fullyear   = 4DIGIT
            date-month      = 2DIGIT  ; 01-12
            date-mday       = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31
            time-hour       = 2DIGIT  ; 00-23
            time-minute     = 2DIGIT  ; 00-59
            time-second     = 2DIGIT  ; 00-58, 00-59, 00-60
            time-secfrac    = "." 1*DIGIT
            time-numoffset  = ("+" / "-") time-hour ":" time-minute
            time-offset     = "Z" / time-numoffset

            partial-time    = time-hour ":" time-minute ":" time-second
                              [time-secfrac]
            full-date       = date-fullyear "-" date-month "-" date-mday
            full-time       = partial-time time-offset

            date-time       = full-date "T" full-time';
        reference "RFC 3339";
    }

    typedef timeticks {
        type uint32;
        description
           "The timeticks type represents a non-negative integer
            which represents the time, modulo 2^32 (4294967296
            decimal), in hundredths of a second between two epochs.
```

```
             When objects are defined which use this type, the
             description of the object identifies both of the reference
             epochs.";
         reference
            "RFC 2578 (STD 58)";
     }

     typedef timestamp {
         type yang:timeticks;
         description
            "The timestamp type represents the value of an associated
             timeticks object at which a specific occurrence
             happened.  The specific occurrence must be defined in the
             description of any object defined using this type.  When
             the specific occurrence occurred prior to the last time
             the associated timeticks attribute was zero, then the
             timestamp value is zero.  Note that this requires all
             timestamp values to be reset to zero when the value of
             the associated timeticks attribute reaches 497+ days and
             wraps around to zero.

             The associated timeticks object must be specified
             in the description of any object using this type.";
         reference
            "RFC 2579 (STD 58)";
     }

     /*
      * collection of generic address types
      */

     typedef phys-address {
         type string;
         description
            "Represents media- or physical-level addresses.";
         reference
            "RFC 2579 (STD 58)";
     }
 }
```

4. Internet Specific Derived Types

```
  module inet-types {

      // XXX namespace to be allocated by IANA

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

      organization
          "YANG Language Design Team";

      contact
          "Juergen Schoenwaelder (Editor)
           <j.schoenwaelder@jacobs-university.de>";

      description
          "This module contains standard derived YANG types
           for Internet addresses and related things.";

      revision 2008-06-07 {
          description "Initial revision.";
      }

      /*
       * collection of protocol field related types
       */

      typedef ip-version {
          type enumeration {
              enum unknown {
                  value 0;
                  description
                     "An unknown or unspecified version of the
                      Internet protocol.";
              }
              enum ipv4 {
                  value 1;
                  description
                     "The IPv4 protocol as defined in RFC 791.";
              }
              enum ipv6 {
                  value 2;
                  description
                     "The IPv6 protocol as defined in RFC 2460.";
              }
          }
          description
```

```
             "This value represents the version of the IP protocol.";
          reference
             "RFC 791 (STD 5), RFC 2460";
      }

      typedef dscp {
          type uint8 {
              range "0..63";
          }
          description
             "The dscp type represents a Differentiated Services
              Code-Point that may be used for marking a traffic
              stream.";
          reference
             "RFC 3289, RFC 2474, RFC 2780";
      }

      typedef flow-label {
          type uint32 {
              range "0..1048575";
          }
          description
             "The flow-label type represents flow identifier or
              Flow Label in an IPv6 packet header that may be
              used to discriminate traffic flows.";
          reference
             "RFC 2460";
      }

      typedef port-number {
          type uint16 {
              range "1..65535";
          }
          description
             "The port-number type represents a 16-bit port
              number of an Internet transport layer protocol
              such as UDP, TCP, DCCP or SCTP. Port numbers are
              assigned by IANA.  A current list of all
              assignments is available from
              <http://www.iana.org/>.

              Note that the value zero is not a valid port
              number. A union type might be used in situations
              where the value zero is meaningful.";
          reference
             "RFC 4001";
      }
```

```
      /*
       * collection of autonomous system related types
       */

      typedef as-number {
          type uint32;
          description
             "The as-number type represents autonomous system numbers
              which identify an Autonomous System (AS). An AS is a set
              of routers under a single technical administration, using
              an interior gateway protocol and common metrics to route
              packets within the AS, and using an exterior gateway
              protocol to route packets to other ASs'. IANA maintains
              the AS number space and has delegated large parts to the
              regional registries.

              Autonomous system numbers are currently limited to 16 bits
              (0..65535). There is however work in progress to enlarge
              the autonomous system number space to 32 bits. This
              textual convention therefore uses an uint32 base type
              without a range restriction in order to support a larger
              autonomous system number space.";
          reference
             "RFC 1771, RFC 1930, RFC 4001";
      }

      /*
       * collection of IP address and hostname related types
       */

      typedef ip-address {
          type union {
              type inet:ipv4-address;
              type inet:ipv6-address;
          }
          description
             "The ip-address type represents an IP address and
              is IP version neutral. The format of the textual
              representations implies the IP version.";
      }

      typedef ipv4-address {
          type string {
              pattern
                 '(([0-1]?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])\.){3}'
               + '([0-1]?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])'
               + '(%[\p{N}\p{L}]+)?';
          }
```

```
          description
             "The ipv4-address type represents an IPv4 address in
              dotted-quad notation. The IPv4 address may include
              a zone index, separated by a % sign.

              The zone index is used to disambiguate identical address
              values.  For link-local addresses, the zone index will
              typically be the interface index number or the name of an
              interface. If the zone index is not present, the default
              zone of the device will be used.";
      }

      typedef ipv6-address {
          type string {
              pattern
                 /* full */
                 '((([0-9a-fA-F]{1,4}:){7})([0-9a-fA-F]{1,4})'
              +  '(%[\p{N}\p{L}]+)?)'
                 /* mixed */
              +  '|((([0-9a-fA-F]{1,4}:){6})(([0-9]{1,3}\.'
              +      '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}))'
              +   '(%[\p{N}\p{L}]+)?)'
                /* shortened */
              + '|((([0-9a-fA-F]{1,4}:)*([0-9a-fA-F]{1,4}))*(::)'
              +   '(([0-9a-fA-F]{1,4}:)*([0-9a-fA-F]{1,4}))*'
              +   '(%[\p{N}\p{L}]+)?)'
                /* shortened mixed */
              + '((([0-9a-fA-F]{1,4}:)*([0-9a-fA-F]{1,4}))*(::)'
              +  '(([0-9a-fA-F]{1,4}:)*([0-9a-fA-F]{1,4}))*'
              +  '(([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}))'
              +   '(%[\p{N}\p{L}]+)?)';
          }
          description
             "The ipv6-address type represents an IPv6 address in
              full, mixed, shortened and shortened mixed notation.
              The IPv6 address may include a zone index, separated
              by a % sign.

              The zone index is used to disambiguate identical address
              values.  For link-local addresses, the zone index will
              typically be the interface index number or the name of an
              interface. If the zone index is not present, the default
              zone of the device will be used.";
          reference
              "RFC 4007: IPv6 Scoped Address Architecture";
      }

      typedef ip-prefix {
```

```
          type union {
              type inet:ipv4-prefix;
              type inet:ipv6-prefix;
          }
          description
             "The ip-prefix type represents an IP prefix and
              is IP version neutral. The format of the textual
              representations implies the IP version.";
      }

      typedef ipv4-prefix {
          type string {
              pattern
                 '(([0-1]?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])\.){3}'
               + '([0-1]?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])'
               + '/\p{N}+';
          }
          description
             "The ipv4-prefix type represents an IPv4 address prefix.
              The prefix length is given by the number following the
              slash character and must be less than or equal 32.

              A prefix length value of n corresponds to an IP address
              mask which has n contiguous 1-bits from the most
              significant bit (MSB) and all other bits set to 0.

              The IPv4 address represented in dotted quad notation
              should have all bits that do not belong to the prefix
              set to zero.";
      }

      typedef ipv6-prefix {
          type string {
              pattern
                 /* full */
                 '((([0-9a-fA-F]{1,4}:){7})([0-9a-fA-F]{1,4})'
              +  '/\p{N}+)'
                 /* mixed */
              +  '|((([0-9a-fA-F]{1,4}:){6})(([0-9]{1,3}\.'
              +      '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}))'
              +   '/\p{N}+)'
                 /* shortened */
              +  '|((([0-9a-fA-F]{1,4}:)*([0-9a-fA-F]{1,4}))*(::)'
              +   '(([0-9a-fA-F]{1,4}:)*([0-9a-fA-F]{1,4}))*'
              +   '/\p{N}+)'
                 /* shortened mixed */
              + '((([0-9a-fA-F]{1,4}:)*([0-9a-fA-F]{1,4}))*(::)'
              +  '(([0-9a-fA-F]{1,4}:)*([0-9a-fA-F]{1,4}))*'
```

```
            +  '(([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}))'
            +   '/\p{N}+)';
        }
        description
           "The ipv6-prefix type represents an IPv6 address prefix.
            The prefix length is given by the number following the
            slash character and must be less than or equal 128.

            A prefix length value of n corresponds to an IP address
            mask which has n contiguous 1-bits from the most
            significant bit (MSB) and all other bits set to 0.

            The IPv6 address should have all bits that do not belong
            to the prefix set to zero.";
    }

    /*
     * Domain name and URI types.
     */

    typedef domain-name {
        type string {
            pattern '([a-zA-Z0-9\-]+\.)*[a-zA-Z0-9\-]+';
        }
        description
           "The domain-name type represents a DNS domain
            name. The name SHOULD be fully qualified
            whenever possible.

            The description clause of objects using the
            domain-name type MUST describe how (and when)
            these names are resolved to IP addresses.

            Note that the resolution of a domain-name value
            may require to query multiple DNS records (e.g.,
            A for IPv4 and AAAA for IPv6).  The order of the
            resolution process and which DNS record takes
            precedence depends on the configuration of the
            resolver.";
        reference
           "RFC 1034";
    }

    typedef host {
        type union {
            type inet:ip-address;
            type inet:domain-name;
        }
```

```
          description
             "The host type represents either an IP address
              or a DNS domain name.";
      }

      typedef uri {
          type string;    // TBD: add the regex from RFC 3986 here?
          description
             "The uri type represents a Uniform Resource Identifier
              (URI) as defined by STD 66.

              Objects using the uri type must be in US-ASCII encoding,
              and MUST be normalized as described by RFC 3986 Sections
              6.2.1, 6.2.2.1, and 6.2.2.2.  All unnecessary
              percent-encoding is removed, and all case-insensitive
              characters are set to lowercase except for hexadecimal
              digits, which are normalized to uppercase as described in
              Section 6.2.2.1.

              The purpose of this normalization is to help provide
              unique URIs.  Note that this normalization is not
              sufficient to provide uniqueness.  Two URIs that are
              textually distinct after this normalization may still be
              equivalent.

              Objects using the uri type may restrict the schemes that
              they permit.  For example, 'data:' and 'urn:' schemes
              might not be appropriate.

              A zero-length URI is not a valid URI.  This can be used to
              express 'URI absent' where required."
          reference "RFC 3986 STD 66 and RFC 3305"
      }

  }
```

5. IEEE 802 Specific Derived Types

```
   module ieee-types {

       // XXX namespace to be allocated by IANA

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

       import yang-types {
           prefix yang;
       }

       organization
           "YANG Language Design Team";

       contact
           "Juergen Schoenwaelder (Editor)
            <j.schoenwaelder@jacobs-university.de>";

       description
           "This module contains standard derived YANG types
            for IEEE 802 addresses and related things.";

       revision 2008-05-22 {
           description "Initial revision.";
       }

       /*
        * collection of IEEE address type definitions
        */

       typedef mac-address {
           type yang:phys-address {
               pattern '([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}';
           }
           description
              "The mac-address type represents an 802 MAC address
               represented in the `canonical' order defined by
               IEEE 802.1a, i.e., as if it were transmitted least
               significant bit first, even though 802.5 (in contrast
               to other 802.x protocols) requires MAC addresses to
               be transmitted most significant bit first.";
           reference
              "RFC 2579 STD 58";
       }

       /*
```

```
        * collection of IEEE 802 related identifier types
        */

       typedef bridgeid {
           type string {
               pattern '[0-9a-fA-F]{4}:'
                     + '([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}';
           }
           description
              "The bridgeid type represents identifers that uniquely
               identify a bridge.  Its first four hexadecimal digits
               contain a priority value followed by a colon. The
               remaining characters contain the MAC address used to
               refer to a bridge in a unique fashion (typically, the
               numerically smallest MAC address of all ports on the
               bridge).";
           reference
              "RFC 4188";
       }

       typedef vlanid {
           type uint16 {
               range "1..4094";
           }
           description
              "The vlanid type uniquely identifies a VLAN. This is
               the 12-bit VLAN-ID used in the VLAN Tag header. The
               range is defined by the referenced specification.";
           reference
              "IEEE Std 802.1Q 2003 Edition, Virtual Bridged Local
               Area Networks.";
       }
   }
```

6. IANA Considerations

A registry for standard YANG modules shall be set up. Each entry shall contain the unique module name, the unique XML namespace from the YANG URI Scheme and some reference to the module's documentation.

This document registers three URIs for the YANG XML namespace in the IETF XML registry [RFC3688].

```
URI: urn:ietf:params:xml:ns:yang:ieee-types

URI: urn:ietf:params:xml:ns:yang:inet-types

URI: urn:ietf:params:xml:ns:yang:yang-types
```

7. Security Considerations

This document defines common data types using the YANG data modeling language. The definitions themselves have no security impact on the Internet but the usage of these definitions in concrete YANG modules might have. The security considerations spelled out in the YANG specification [YANG] apply for this document as well.

8. Contributors

The following people all contributed significantly to the initial version of this draft:

- Andy Bierman (andybierman.com)
- Martin Bjorklund (Tail-f Systems)
- Balazs Lengyel (Ericsson)
- David Partain (Ericsson)
- Phil Shafer (Juniper Networks)

9. Open Issues

- Should YANG allow multiple pattern that get ANDed? This would for example allow to tighten the IPv6 pattern.

  Message-Id: <1215432618.23783.59.camel@missotis>

- Add some common reusable groupings, e.g. a combination of ip-address and port-number? Or should such groupings be a separate document?

10. References

10.1. Normative References

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

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

[YANG] Bjorklund, M., Ed., "YANG - A data modeling language for NETCONF", draft-ietf-netmod-yang-00 (work in progress).

10.2. Informative References

[RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

Author's Address

Juergen Schoenwaelder (editor)
Jacobs University

Email: j.schoenwaelder@jacobs-university.de

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).