NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 31, 2019

K. Watsen
Watsen Networks

April 29, 2019

YANG Data Model for a Centralized Keystore Mechanism
draft-ietf-netconf-keystore-09

Abstract

This document defines a YANG 1.1 module called "ietf-keystore" that enables centralized configuration of asymmetric keys and their associated certificates, and notification for when configured certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

- "VVVV" --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

- "2019-04-29" --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

- Appendix A. Change Log

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG 1.1 [RFC7950] module called "ietf-keystore" that enables centralized configuration of asymmetric keys and their associated certificates, and notification for when configured certificates are about to expire.

This module also defines Six groupings designed for maximum reuse. These groupings include one for the public half of an asymmetric key, one for both the public and private halves of an asymmetric key, one for both halves of an asymmetric key and a list of associated certificates, one for an asymmetric key that may be configured locally or via a reference to an asymmetric key in the keystore, one for a trust anchor certificate and, lastly, one for an end entity certificate.

Special consideration has been given for systems that have cryptographic hardware, such as a Trusted Protection Module (TPM). These systems are unique in that the cryptographic hardware completely hides the private keys and must perform all private key operations. To support such hardware, the "private-key" can be the special value "permanently-hidden" and the actions "generate-hidden-key" and "generate-certificate-signing-request" can be used to direct these operations to the hardware .

This document in compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, to support keys and associated certificates installed during manufacturing (e.g., for a IDevID [Std-802.1AR-2009] certificate), it is expected that such data may appear only in <operational>.

While only asymmetric keys are currently supported, the module has been designed to enable other key types to be introduced in the future.

The module does not support protecting the contents of the keystore (e.g., via encryption), though it could be extended to do so in the future.

It is not required that a system has an operating system level keystore utility to implement this module.

2. Requirements Language

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. The Keystore Model

3.1. Tree Diagram

This section provides a tree diagrams [RFC8340] for the "ietf-keystore" module that presents both the protocol-accessible "keystore" as well the all the groupings intended for external usage.

```
module: ietf-keystore
  +--rw keystore
     +--rw asymmetric-keys
        +--rw asymmetric-key* [name]
           +--rw name                                    string
           +--rw algorithm?
           |       asymmetric-key-algorithm-ref
           +--rw public-key?                             binary
           +--rw private-key?                            union
           +---x generate-hidden-key
           |  +---w input
           |     +---w algorithm    asymmetric-key-algorithm-ref
           +---x install-hidden-key
```

```
            |  +---w input
            |     +---w algorithm      asymmetric-key-algorithm-ref
            |     +---w public-key?    binary
            |     +---w private-key?   binary
            +--rw certificates
            |  +--rw certificate* [name]
            |     +--rw name                       string
            |     +--rw cert?                      end-entity-cert-cms
            |     +---n certificate-expiration
            |        +-- expiration-date    yang:date-and-time
            +---x generate-certificate-signing-request
               +---w input
               |  +---w subject       binary
               |  +---w attributes?   binary
               +--ro output
                  +--ro certificate-signing-request    binary

  grouping local-or-keystore-asymmetric-key-grouping
    +-- (local-or-keystore)
```



```
          +--:(local) {local-keys-supported}?
          |  +-- local-definition
          |     +-- algorithm?             asymmetric-key-algorithm-ref
          |     +-- public-key?            binary
          |     +-- private-key?           union
          |     +---x generate-hidden-key
          |     |  +---w input
          |     |     +---w algorithm    asymmetric-key-algorithm-ref
          |     +---x install-hidden-key
          |        +---w input
          |           +---w algorithm      asymmetric-key-algorithm-ref
          |           +---w public-key?    binary
          |           +---w private-key?   binary
          +--:(keystore) {keystore-supported}?
             +-- keystore-reference?   ks:asymmetric-key-ref
  grouping local-or-keystore-asymmetric-key-with-certs-grouping
    +-- (local-or-keystore)
       +--:(local) {local-keys-supported}?
       |  +-- local-definition
       |     +-- algorithm?
       |     |       asymmetric-key-algorithm-ref
       |     +-- public-key?                              binary
```

```
|     +-- private-key?                              union
|     +---x generate-hidden-key
|     |  +---w input
|     |     +---w algorithm    asymmetric-key-algorithm-ref
|     +---x install-hidden-key
|     |  +---w input
|     |     +---w algorithm      asymmetric-key-algorithm-ref
|     |     +---w public-key?    binary
|     |     +---w private-key?   binary
|     +-- certificates
|     |  +-- certificate* [name]
|     |     +-- name?                      string
|     |     +-- cert?                      end-entity-cert-cms
|     |     +---n certificate-expiration
|     |        +-- expiration-date    yang:date-and-time
|     +---x generate-certificate-signing-request
|        +---w input
|        |  +---w subject       binary
|        |  +---w attributes?   binary
|        +--ro output
|           +--ro certificate-signing-request    binary
|  +--:(keystore) {keystore-supported}?
|     +-- keystore-reference?   ks:asymmetric-key-ref
grouping local-or-keystore-end-entity-cert-with-key-grouping
  +-- (local-or-keystore)
     +--:(local) {local-keys-supported}?
```



```
|  +-- local-definition
|     +-- algorithm?
|     |       asymmetric-key-algorithm-ref
|     +-- public-key?               binary
|     +-- private-key?              union
|     +---x generate-hidden-key
|     |  +---w input
|     |     +---w algorithm    asymmetric-key-algorithm-ref
|     +---x install-hidden-key
|     |  +---w input
|     |     +---w algorithm      asymmetric-key-algorithm-ref
|     |     +---w public-key?    binary
|     |     +---w private-key?   binary
|     +-- cert?                     end-entity-cert-cms
|     +---n certificate-expiration
```

```
          |        +-- expiration-date    yang:date-and-time
          +--:(keystore) {keystore-supported}?
             +-- keystore-reference?   ks:asymmetric-key-certificate-ref
```

3.2. Example Usage

The following example illustrates what a fully configured keystore might look like in <operational>, as described by Section 5.3 in [RFC8342]. This datastore view illustrates data set by the manufacturing process alongside conventional configuration. This keystore instance has four keys, two having one associated certificate, one having two associated certificates, and one empty key.

```
=========== NOTE: '\' line wrapping per BCP XX (RFC XXXX) ===========

<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
          xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
          xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
          or:origin="or:intended">
  <asymmetric-keys>

    <asymmetric-key>
      <name>ex-rsa-key</name>
      <algorithm>ct:rsa2048</algorithm>
      <private-key>base64encodedvalue==</private-key>
      <public-key>base64encodedvalue==</public-key>
      <certificates>
        <certificate>
          <name>ex-rsa-cert</name>
          <cert>base64encodedvalue==</cert>
        </certificate>
      </certificates>
```



```
    </asymmetric-key>

<!-- waiting for Haiguang fix...
    <asymmetric-key>
      <name>tls-ec-key</name>
      <algorithm>ct:secp256r1</algorithm>
      <private-key>base64encodedvalue==</private-key>
      <public-key>base64encodedvalue==</public-key>
```

```
        <certificates>
          <certificate>
            <name>tls-ec-cert</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>
  -->

      <asymmetric-key>
        <name>tpm-protected-key</name>
        <algorithm or:origin="or:system">ct:rsa2048</algorithm>
        <private-key or:origin="or:system">permanently-hidden</private\
  -key>
        <public-key or:origin="or:system">base64encodedvalue==</public\
  -key>
        <certificates>
          <certificate or:origin="or:system">
            <name>builtin-idevid-cert</name>
            <cert or:origin="or:system">base64encodedvalue==</cert>
          </certificate>
          <certificate>
            <name>my-ldevid-cert</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>

      <asymmetric-key>
        <name>tpm-protected-key2</name>
        <certificates>
          <certificate>
            <name>builtin-idevid-cert2</name>
          </certificate>
          <certificate>
            <name>my-ldevid-cert2</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>
```



```
    </asymmetric-keys>
```

```
</keystore>
```

The following example module has been constructed to illustrate the "local-or-keystore-asymmetric-key-grouping" grouping defined in the "ietf-keystore" module.

```
module ex-keystore-usage {
  yang-version 1.1;

  namespace "http://example.com/ns/example-keystore-usage";
  prefix "eku";

  import ietf-keystore {
    prefix ks;
    reference
      "RFC VVVV: YANG Data Model for a 'Keystore' Mechanism";
  }

  organization
   "Example Corporation";

  contact
   "Author: YANG Designer <mailto:yang.designer@example.com>";

  description
   "This module illustrates the grouping in the keystore draft called
    'local-or-keystore-asymmetric-key-with-certs-grouping'.";

  revision "YYYY-MM-DD" {
    description
     "Initial version";
    reference
     "RFC XXXX: YANG Data Model for a 'Keystore' Mechanism";
  }

  container keystore-usage {
    description
      "An illustration of the various keystore groupings.";

    list just-a-key {
      key name;
      leaf name {
        type string;
        description
          "An arbitrary name for this key.";
      }
```

```
        uses ks:local-or-keystore-asymmetric-key-grouping;
        description
          "An asymmetric key, with no certs, that may be configured
           locally or be a reference to an asymmetric key in the
           keystore.  The intent is to reference just the asymmetric
           key, not any certificates that may also be associated
           with the asymmetric key.";
      }

      list key-with-certs {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-asymmetric-key-with-certs-grouping;
        description
          "An asymmetric key and its associated certs, that may be
           configured locally or be a reference to an asymmetric key
           (and its associated certs) in the keystore.";
      }

      list end-entity-cert-with-key {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
        description
          "An end-entity certificate, and its associated private key,
           that may be configured locally or be a reference to a
           specific certificate (and its associated private key) in
           the keystore.";
      }
    }

  }
```

The following example illustrates what two configured keys, one local and the other remote, might look like. This example consistent with other examples above (i.e., the referenced key is in an example above).

```
   <keystore-usage xmlns="http://example.com/ns/example-keystore-usage">

     <!-- ks:local-or-keystore-asymmetric-key-grouping -->

     <just-a-key>
       <name>a locally-defined key</name>
       <local-definition>
         <algorithm
           xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
           ct:rsa2048
         </algorithm>
         <private-key>base64encodedvalue==</private-key>
         <public-key>base64encodedvalue==</public-key>
       </local-definition>
     </just-a-key>

     <just-a-key>
       <name>a keystore-defined key (and its associated certs)</name>
       <keystore-reference>ex-rsa-key</keystore-reference>
     </just-a-key>

     <!-- ks:local-or-keystore-key-and-end-entity-cert-grouping -->

     <key-with-certs>
       <name>a locally-defined key with certs</name>
       <local-definition>
         <algorithm
           xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
           ct:rsa2048
         </algorithm>
         <private-key>base64encodedvalue==</private-key>
         <public-key>base64encodedvalue==</public-key>
         <certificates>
           <certificate>
             <name>a locally-defined cert</name>
             <cert>base64encodedvalue==</cert>
           </certificate>
         </certificates>
       </local-definition>
```

```
    </key-with-certs>

    <key-with-certs>
      <name>a keystore-defined key (and its associated certs)</name>
      <keystore-reference>ex-rsa-key</keystore-reference>
    </key-with-certs>

    <!-- ks:local-or-keystore-end-entity-cert-with-key-grouping -->
```



```
    <end-entity-cert-with-key>
      <name>a locally-defined end-entity cert with key</name>
      <local-definition>
        <algorithm
          xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
          ct:rsa2048
        </algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
        <cert>base64encodedvalue==</cert>
      </local-definition>
    </end-entity-cert-with-key>

    <end-entity-cert-with-key>
      <name>a keystore-defined certificate (and its associated key)</n\
  ame>
      <keystore-reference>ex-rsa-cert</keystore-reference>
    </end-entity-cert-with-key>

  </keystore-usage>
```

3.3. YANG Module

This YANG module has normative references to [RFC8341] and [I-D.ietf-netconf-crypto-types], and an informative reference to [RFC8342].

```
<CODE BEGINS> file "ietf-keystore@2019-04-29.yang"
module ietf-keystore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-keystore";
  prefix ks;
```

```
  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC CCCC: Common YANG Data Types for Cryptography";
  }

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
```



```
  contact
    "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>
     Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

  description
    "This module defines a keystore to centralize management
     of security credentials.

     Copyright (c) 2019 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
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC
     itself for full legal notices.;

     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 (RFC 2119)
       (RFC 8174) when, and only when, they appear in all
       capitals, as shown here.";

     revision 2019-04-29 {
       description
         "Initial version";
       reference
         "RFC VVVV:
            YANG Data Model for a Centralized Keystore Mechanism";
     }

     /****************/
     /*   Features   */
     /****************/

     feature keystore-supported {
       description
         "The 'keystore-supported' feature indicates that the server
          supports the keystore.";
     }
```



```
     feature local-keys-supported {
       description
         "The 'local-keys-supported' feature indicates that the
          server supports locally-defined keys.";
     }

     /****************/
     /*   Typedefs   */
     /****************/

     typedef asymmetric-key-ref {
       type leafref {
         path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
            + "/ks:name";
       }
       description
         "This typedef enables modules to easily define a reference
          to an asymmetric key stored in the keystore.";
```

```
    reference
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }

  typedef asymmetric-key-certificate-ref {
    type leafref {
      path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
         + "/ks:certificates/ks:certificate/ks:name";
    }
    description
      "This typedef enables modules to easily define a reference
       to a specific certificate associated with an asymmetric key
       stored in the keystore.";
    reference
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }

  /*****************/
  /*   Groupings   */
  /*****************/

  grouping local-or-keystore-asymmetric-key-grouping {
    description
      "A grouping that expands to allow the asymmetric key to be
       either stored locally, within the using data model, or be
       a reference to an asymmetric key stored in the keystore.";
    choice local-or-keystore {
      mandatory true;
      case local {
        if-feature "local-keys-supported";
```



```
        container local-definition {
          must '(algorithm and public-key and private-key)
                or not (algorithm or public-key or private-key)' {
            description
              "These descendent nodes are not mandatory because they
               MAY be defined in <operational>.  Implementations MUST
               assert that these values are either configured or that
               they exist in <operational>.";
          }
          description
            "Container to hold the local key definition.";
```

```
          uses ct:asymmetric-key-pair-grouping;
        }
      }
      case keystore {
        if-feature "keystore-supported";
        leaf keystore-reference {
          type ks:asymmetric-key-ref;
          description
            "A reference to an asymmetric key that exists in
             the keystore.  The intent is to reference just the
             asymmetric key, not any certificates that may also
             be associated with the asymmetric key.";
        }
      }
      description
        "A choice between an inlined definition and a definition
         that exists in the keystore.";
    }
  }

  grouping local-or-keystore-asymmetric-key-with-certs-grouping {
    description
      "A grouping that expands to allow an asymmetric key and its
       associated certificates to be either stored locally, within
       the using data model, or be a reference to an asymmetric key
       (and its associated certificates) stored in the keystore.";
    choice local-or-keystore {
      mandatory true;
      case local {
        if-feature "local-keys-supported";
        container local-definition {
          must '(algorithm and public-key and private-key)
                or not (algorithm or public-key or private-key)' {
            description
              "These descendent nodes are not mandatory because they
               MAY be defined in <operational>.  Implementations MUST
               assert that these values are either configured or that
```



```
               they exist in <operational>.";
          }
          description
            "Container to hold the local key definition.";
```

```
          uses ct:asymmetric-key-pair-with-certs-grouping;
        }
      }
      case keystore {
        if-feature "keystore-supported";
        leaf keystore-reference {
          type ks:asymmetric-key-ref;
          description
            "A reference to a value that exists in the keystore.";
        }
      }
      description
        "A choice between an inlined definition and a definition
         that exists in the keystore.";
    }
  }

  grouping local-or-keystore-end-entity-cert-with-key-grouping {
    description
      "A grouping that expands to allow an end-entity certificate
       (and its associated private key) to be either stored locally,
       within the using data model, or be a reference to a specific
       certificate in the keystore.";
    choice local-or-keystore {
      mandatory true;
      case local {
        if-feature "local-keys-supported";
        container local-definition {
          must '(algorithm and public-key and private-key)
                or not (algorithm or public-key or private-key)' {
            description
              "These descendent nodes are not mandatory because they
               MAY be defined in <operational>.  Implementations MUST
               assert that these values are either configured or that
               they exist in <operational>.";
          }
          description
            "Container to hold the local key definition.";
          uses ct:asymmetric-key-pair-grouping;
          uses ct:end-entity-cert-grouping;
        }
      }
      case keystore {
        if-feature "keystore-supported";
```

```
          leaf keystore-reference {
            type ks:asymmetric-key-certificate-ref;
            description
              "A reference to a specific certificate, and its
               associated private key, stored in the keystore.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
      }
    }

    /*********************************/
    /*   Protocol accessible nodes   */
    /*********************************/


    container keystore {
      nacm:default-deny-write;
      description
        "The keystore contains a list of keys.";
      container asymmetric-keys {
        description
          "A list of asymmetric keys.";
        list asymmetric-key {
          must '(algorithm and public-key and private-key)
                or not (algorithm or public-key or private-key)' {
            description
              "These descendent nodes are not mandatory because they
               MAY be defined in <operational>.  Implementations MUST
               assert that these values are either configured or that
               they exist in <operational>.";
          }
          key "name";
          description
            "An asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the asymmetric key.  If the name
               matches the name of a key that exists independently in
               <operational> (i.e., a 'permanently-hidden' key), then
               the 'algorithm', 'public-key', and 'private-key' nodes
               MUST NOT be configured.";
          }
          uses ct:asymmetric-key-pair-with-certs-grouping;
```

```
        }
```



```
      }
    }
  }
  <CODE ENDS>
```

4. Security Considerations

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

There are a number of data nodes defined in this YANG module that 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:

/: The entire data tree defined by this module is sensitive to write operations. For instance, the addition or removal of keys, certificates, etc., can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for the entire data tree.

/keystore/asymmetric-keys/asymmetric-key/private-key: When writing this node, implementations MUST ensure that the strength of the key being configured is not greater than the strength of the underlying secure transport connection over which it is communicated. Implementations SHOULD fail the write-request if ever the strength of the private key is greater then the strength of the underlying transport, and alert the client that the strength of the key may have been

compromised. Additionally, when deleting this node, implementations SHOULD automatically (without explicit request) zeroize these keys in the most secure manner available, so as to prevent the remnants of their persisted storage locations from being analyzed in any meaningful way.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus

important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/keystore/asymmetric-keys/asymmetric-key/private-key: This node is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. The best reason for returning this node is to support backup/restore type workflows. For this reason, the NACM extension "default-deny-all" has been set for this data node.

5. IANA Considerations

5.1. The IETF XML Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

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

5.2. The YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the the following registration is requested:

```
name:         ietf-keystore
namespace:    urn:ietf:params:xml:ns:yang:ietf-keystore
prefix:       ks
reference:    RFC VVVV
```

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-05 (work in progress), March 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

6.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF

Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[Std-802.1AR-2009] Group, W. -. H. L. L. P. W., "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

Appendix A. Change Log

A.1. 00 to 01

- Replaced the 'certificate-chain' structures with PKCS#7 structures. (Issue #1)
- Added 'private-key' as a configurable data node, and removed the 'generate-private-key' and 'load-private-key' actions. (Issue #2)
- Moved 'user-auth-credentials' to the ietf-ssh-client module. (Issues #4 and #5)

A.2. 01 to 02

- Added back 'generate-private-key' action.
- Removed 'RESTRICTED' enum from the 'private-key' leaf type.

- Fixed up a few description statements.

A.3. 02 to 03

- Changed draft's title.
- Added missing references.
- Collapsed sections and levels.
- Added RFC 8174 to Requirements Language Section.
- Renamed 'trusted-certificates' to 'pinned-certificates'.
- Changed 'public-key' from config false to config true.
- Switched 'host-key' from OneAsymmetricKey to definition from RFC 4253.

A.4. 03 to 04

- Added typedefs around leafrefs to common keystore paths
- Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- Removed Design Considerations section
- Moved key and certificate definitions from data tree to groupings

A.5. 04 to 05

- Removed trust anchors (now in their own draft)
- Added back global keystore structure
- Added groupings enabling keys to either be locally defined or a reference to the keystore.

A.6. 05 to 06

- Added feature "local-keys-supported"

- Added nacm:default-deny-all and nacm:default-deny-write
- Renamed generate-asymmetric-key to generate-hidden-key
- Added an install-hidden-key action
- Moved actions inside fo the "asymmetric-key" container
- Moved some groupings to draft-ietf-netconf-crypto-types

A.7. 06 to 07

- Removed a "require-instance false"
- Clarified some description statements
- Improved the keystore-usage examples

A.8. 07 to 08

- Added "local-definition" containers to avoid posibility of the action/notification statements being under a "case" statement.
- Updated copyright date, boilerplate template, affiliation, folding algorithm, and reformatted the YANG module.

A.9. 08 to 09

- Added a 'description' statement to the 'must' in the /keystore/asymmetric-key node explaining that the descendent values may exist in <operational> only, and that implementation MUST assert that the values are either configured or that they exist in <operational>.

- Copied above 'must' statement (and description) into the local-or-keystore-asymmetric-key-grouping, local-or-keystore-asymmetric-key-with-certs-grouping, and local-or-keystore-end-entity-cert-with-key-grouping statements.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Ramkumar Dhanapal, Mehmet Ersue, Balazs Kovacs, David Lamparter, Ladislav Lhotka, Alan Luchuk, Mahesh Jethanandani, Radek Krejci, Reshad Rahman, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, Eric Voit, Bert Wijnen, and Liang Xia.

Author's Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net