Abstract

This document defines a YANG 1.1 module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted. Asymmetric keys may be associated with certificates. Notifications are sent when 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:

- "2019-07-02" --> 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/.
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 3, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 3
2. Requirements Language ....................................... 3
3. The Keystore Model .......................................... 4
   3.1. Tree Diagram ........................................... 4
   3.2. Example Usage .......................................... 12
       3.2.1. A Keystore Instance ............................... 12
       3.2.2. The "generate-symmetric-key" RPC ............... 14
       3.2.3. Notable Keystore Groupings .................... 14
   3.3. YANG Module ........................................... 18
4. Security Considerations ...................................... 27
5. IANA Considerations ......................................... 28
   5.1. The IETF XML Registry .................................. 29
   5.2. The YANG Module Names Registry ...................... 29
6. References .................................................... 29
   6.1. Normative References ................................... 29
   6.2. Informative References ................................. 30
Appendix A. Change Log ........................................... 31
   A.1. 00 to 01 .................................................. 31
   A.2. 01 to 02 .................................................. 31
   A.3. 02 to 03 .................................................. 31
   A.4. 03 to 04 .................................................. 31
   A.5. 04 to 05 .................................................. 32
   A.6. 05 to 06 .................................................. 32
   A.7. 06 to 07 .................................................. 32
This document defines a YANG 1.1 [RFC7950] module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

The "ietf-keystore" module defines many "grouping" statements intended for use by other modules that may import it. For instance, there are groupings that defined enabling a key to be either configured locally (within the defining data model) or be a reference to a key in the keystore.

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 hides the secret key values. To support such hardware, symmetric keys may have the value "hidden-key" and asymmetric keys may have the value "hidden-private-key". While how such keys are created or destroyed is outside the scope of this document, the keystore can contain entries for such keys, enabling them to be reference by other configuration elements.

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

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
          +--rw asymmetric-key-algorithm-t
        +--rw public-key binary
        +--rw (private-key-type)
          +--:(private-key)
            +--rw private-key? binary
            +--:(hidden-private-key)
              +--rw hidden-private-key? empty
            +--:(encrypted-private-key)
              +--rw encrypted-private-key
                +--rw (key-type)
                  +--:(symmetric-key-ref)
                    +--rw symmetric-key-ref? leafref
                      {keystore-supported}?
                  +--:(asymmetric-key-ref)
                    +--rw asymmetric-key-ref? leafref
                      {keystore-supported}?
                +--rw value? 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
  +--rw symmetric-keys
    +--rw symmetric-key* [name]
      +--rw name string
      +--rw algorithm encryption-algorithm-t
      +--rw (key-type)
++-:(key)
  |  +-rw key?     binary
++-:(hidden-key)
  |  +-rw hidden-key?     empty
++-:(encrypted-key)
  +-rw encrypted-key
    +-rw (key-type)
      +-rw symmetric-key-ref?
        leafref
          {keystore-supported}?
      +-:(asymmetric-key-ref)
        +-rw asymmetric-key-ref?
          leafref
            {keystore-supported}?
          value?     binary

rpcs:
  +++-x generate-symmetric-key
    +++-w input
      |        +-w algorithm     ct:encryption-algorithm-t
      |        +-w encrypt-with!
        +-rw (key-type)
          +-rw symmetric-key-ref?
            leafref
              {keystore-supported}?
          +-:(asymmetric-key-ref)
            +-rw asymmetric-key-ref?
              leafref
                {keystore-supported}?
            value?     binary
    +++-ro output
      +%ro algorithm     encryption-algorithm-t
      +%ro (key-type)
        %ro (key)
          %ro key?     binary
        %ro (hidden-key)
          %ro hidden-key?     empty
        %ro (encrypted-key)
          %ro encrypted-key
            %ro (key-type)
              %ro symmetric-key-ref?
                leafref
                  {keystore-supported}?
              %ro asymmetric-key-ref?
                leafref
                  {keystore-supported}?
              value?     binary
  +++-x generate-asymmetric-key
    +++-w input
      |        +-w algorithm     ct:asymmetric-key-algorithm-t
      |        +-w encrypt-with!

<pre>```yang
++-w (key-type)
  +--:(symmetric-key-ref)
      |  ++-w symmetric-key-ref? leafref
      |      (keystore-supported)?
      +--:(asymmetric-key-ref)
          |  ++-w asymmetric-key-ref? leafref
          |      (keystore-supported)?
  +--ro output
      |  +++-ro algorithm
      |      asymmetric-key-algorithm-t
      +--ro public-key binary
      +--ro (private-key-type)
          +--:(private-key)
              |  +++-ro private-key? binary
              +--:(hidden-private-key)
                  |  +++-ro hidden-private-key? empty
                  +--:(encrypted-private-key)
                      |  +++-ro encrypted-private-key
                          +--ro (key-type)
                              |  +++-ro symmetric-key-ref? leafref
                              |      (keystore-supported)?
                              +--:(asymmetric-key-ref)
                                  |  +++-ro asymmetric-key-ref? leafref
                                  |      (keystore-supported)?
                      +--ro value? binary

grouping key-reference-type-grouping
  +-- (key-type)
      +--:(symmetric-key-ref)
          |  ++ symmetric-key-ref?
          |      -> /keystore/symmetric-keys/symmetric-key/name
          |      (keystore-supported)?
      +--:(asymmetric-key-ref)
          |  ++ asymmetric-key-ref?
          |      -> /keystore/asymmetric-keys/asymmetric-key/name
          |      (keystore-supported)?

groupBy encrypted-value-grouping
  +-- (key-type)
      +--:(symmetric-key-ref)
          |  ++ symmetric-key-ref?
          |      -> /keystore/symmetric-keys/symmetric-key/name
          |      (keystore-supported)?
      +--:(asymmetric-key-ref)
          |  ++ asymmetric-key-ref?
          |      -> /keystore/asymmetric-keys/asymmetric-key/name
          |      (keystore-supported)?
      +-- ro value? binary
```</pre>
grouping symmetric-key-grouping
  +-- algorithm              encryption-algorithm-t
  +-- (key-type)
    +--:(key)
      |  +-- key?             binary
    +--:(hidden-key)
      |  +-- hidden-key?     empty
    +--:(encrypted-key)
      +-- encrypted-key
        +-- (key-type)
          |  +--:(symmetric-key-ref)
          |    |  +-- symmetric-key-ref?     leafref
          |    |         (keystore-supported)?
          |  +--:(asymmetric-key-ref)
          |    +-- asymmetric-key-ref?    leafref
          |         (keystore-supported)?
          +-- value?                      binary

grouping asymmetric-key-pair-grouping
  +-- algorithm              asymmetric-key-algorithm-t
  +-- public-key                     binary
  +-- (private-key-type)
    +--:(private-key)
      |  +-- private-key?             binary
    +--:(hidden-private-key)
      |  +-- hidden-private-key?     empty
    +--:(encrypted-private-key)
      +-- encrypted-private-key
        +-- (key-type)
          |  +--:(symmetric-key-ref)
          |    |  +-- symmetric-key-ref?     leafref
          |    |         (keystore-supported)?
          |  +--:(asymmetric-key-ref)
          |    +-- asymmetric-key-ref?    leafref
          |         (keystore-supported)?
          +-- value?                      binary

grouping asymmetric-key-pair-with-cert-grouping
  +-- algorithm
    |    asymmetric-key-algorithm-t
  +-- public-key                     binary
  |    (private-key-type)
  |    +--:(private-key)
  |      |  +-- private-key?             binary
  |    +--:(hidden-private-key)
  |      |  +-- hidden-private-key?     empty
  |    +--:(encrypted-private-key)
  |      +-- encrypted-private-key
  |        +-- (key-type)
  |          |  +--:(symmetric-key-ref)
grouping asymmetric-key-pair-with-certs-grouping
  +-- algorithm
  |       asymmetric-key-algorithm-t
  +-- public-key
  +-- (private-key-type)
  |  +-- private-key
  |  +-- (hidden-private-key)
  |  +-- (encrypted-private-key)
  +-- encrypted-private-key
     +-- (key-type)
     |  +-- (symmetric-key-ref)
     |  |  +-- symmetric-key-ref
     |  |     (keystore-supported)?
     |  +-- (asymmetric-key-ref)
     |     +-- asymmetric-key-ref
     |     |     (keystore-supported)?
     |     +-- value
     |          binary
     +-- cert?
          end-entity-cert-cms
     +--n certificate-expiration
     |  +-- expiration-date
     |     yang:date-and-time
     +--x generate-certificate-signing-request
        +--w input
        |  +--w subject
        |  +--w attributes
        +--ro output
        +--ro certificate-signing-request
     grouping asymmetric-key-pair-with-certs-grouping
     +-- algorithm
     |       asymmetric-key-algorithm-t
     +-- public-key
     +-- (private-key-type)
     |  +-- private-key
     |  +-- (hidden-private-key)
     |  +-- (encrypted-private-key)
     +-- encrypted-private-key
        +-- (key-type)
        |  +-- (symmetric-key-ref)
        |  |  +-- symmetric-key-ref
        |  |     (keystore-supported)?
        |  +-- (asymmetric-key-ref)
        |     +-- asymmetric-key-ref
        |     |     (keystore-supported)?
        |     +-- value
        |          binary
     +-- certificates
     |  +-- certificate* [name]
     |     +-- name
     |     +-- cert
          end-entity-cert-cms
     |     +--n certificate-expiration
          +-- expiration-date
              yang:date-and-time
     +--x generate-certificate-signing-request
        +--w input
        |  +--w subject
        |  +--w attributes
        +--ro output
        +--ro certificate-signing-request
     grouping asymmetric-key-certificate-ref-grouping
     +-- asymmetric-key
          ks:asymmetric-key-ref
-- certificate?  leafref
grouping local-or-keystore-asymmetric-key-grouping
  -- (local-or-keystore)
  +-- (local) {local-definitions-supported}?
    -- local-definition
    +-- algorithm
      |  asymmetric-key-algorithm-t
      +-- public-key  binary
    +-- (private-key-type)
      +-- (private-key)
        |  +-- private-key?  binary
        +-- (hidden-private-key)
          |  +-- hidden-private-key?  empty
          +-- (encrypted-private-key)
            +-- encrypted-private-key
              +-- (key-type)
                +-- (symmetric-key-ref)
                  |  +-- symmetric-key-ref?  leafref
                  |    {keystore-supported}?
                  +-- (asymmetric-key-ref)
                    +-- asymmetric-key-ref?  leafref
                    {keystore-supported}?
                +-- value?  binary
      +-- (keystore) {keystore-supported}?
        +-- keystore-reference?  ks:asymmetric-key-ref
  grouping local-or-keystore-asymmetric-key-with-certs-grouping
  +-- (local-or-keystore)
  +-- (local) {local-definitions-supported}?
    -- local-definition
    +-- algorithm
      |  asymmetric-key-algorithm-t
      +-- public-key  binary
    +-- (private-key-type)
      +-- (private-key)
        |  +-- private-key?  binary
        +-- (hidden-private-key)
          |  +-- hidden-private-key?  empty
          +-- (encrypted-private-key)
            +-- encrypted-private-key
              +-- (key-type)
                +-- (symmetric-key-ref)
                  |  +-- symmetric-key-ref?  leafref
                  |    {keystore-supported}?
                  +-- (asymmetric-key-ref)
                    +-- asymmetric-key-ref?  leafref
                    {keystore-supported}?
              +-- value?  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
++-- (local-or-keystore)
      +++-- (local) {local-definitions-supported}?
      ++-- local-definition
         ++-- algorithm
           |    asymmetric-key-algorithm-t
         ++-- public-key binary
         +++-- (private-key-type)
            | ++-- (private-key)
            |    | ++-- private-key? binary
            | +++-- (private-key)
            |    | ++-- (hidden-private-key)
            |    |    | ++-- hidden-private-key? empty
            | +++-- (encrypted-private-key)
            | +++-- encrypted-private-key
            |    | +++-- (key-type)
            |    |    | +++-- (symmetric-key-ref)
            |    |    |    | ++-- symmetric-key-ref? leafref
            |    |    |    |    | (keystore-supported)?
            |    | +++-- (asymmetric-key-ref)
            |    |    | ++-- asymmetric-key-ref? leafref
            |    |    | (keystore-supported)?
            |    | ++-- value? binary
         ++-- 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?
   ++-- asymmetric-key? ks:asymmetric-key-ref

Watsen                   Expires January 3, 2020               [Page 10]
++-- certificate?     leafref
++-- asymmetric-keys
++-- asymmetric-key* [name]
    ++-- name?                      string
    ++-- algorithm
        |  asymmetric-key-algorithm-t
    ++-- public-key
        |  (private-key-type)
        |     ++-- (private-key)
        |        |  ++-- private-key?     binary
        |        |  ++-- (hidden-private-key)
        |        |        |  ++-- hidden-private-key?
        |        |  ++-- (encrypted-private-key)
        |        |        |     ++-- encrypted-private-key
        |        |        |             ++-- (key-type)
        |        |        |             |  ++-- (symmetric-key-ref)
        |        |        |             |     ++-- symmetric-key-ref?    leafref
        |        |        |             |     |  (keystore-supported)?
        |        |        |             |     ++-- (asymmetric-key-ref)
        |        |        |             |     |  ++-- asymmetric-key-ref?    leafref
        |        |        |             |     |  |  (keystore-supported)?
        |        |        |             |     |     ++-- value?        binary
    ++-- certificates
    ++-- certificate* [name]
        |  ++-- name?             string
        |  ++-- cert?             end-entity-cert-cms
        |       ++-- certificate-expiration
        |           ++-- expiration-date    yang:date-and-time
    +++--x generate-certificate-signing-request
        +++--w input
        |        |  ++--w subject     binary
        |        |  ++--w attributes?  binary
        +++--x output
            +++--x certificate-signing-request    binary
    ++-- symmetric-keys
    ++-- symmetric-key* [name]
        |  ++-- name?            string
        |  ++-- algorithm
            |  encryption-algorithm-t
        |  (key-type)
        |     ++-- (key)
        |     |  ++-- key?             binary
        |     |  ++-- (hidden-key)
        |     |        |  ++-- hidden-key?
        |     |  ++-- (encrypted-key)
        |     |        |     ++-- encrypted-key
        |     |        |        |  ++-- (key-type)
    |     |        |     |     |  ++-- (symmetric-key-ref)
3.2. Example Usage

3.2.1. A Keystore Instance

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.

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  or:origin="or:intended">

<!-- Asymmetric Keys -->
<asymmetric-keys>

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

<asymmetric-key>
  <name>tls-ec-key</name>
  <algorithm>secp256r1</algorithm>
  <public-key>base64encodedvalue==</public-key>
  <private-key>base64encodedvalue==</private-key>
  <certificates>
    <certificate>
      <name>ex-rsa-cert</name>
      <cert>base64encodedvalue==</cert>
    </certificate>
  </certificates>
</asymmetric-key>

```

NOTE: '
’ line wrapping per BCP XX (RFC XXXX)
<asymmetric-key>
  <name>tls-ec-cert</name>
  <cert>base64encodedvalue==</cert>
</asymmetric-key>

<asymmetric-key>
  <name>tpm-protected-key</name>
  <algorithm>rsa2048</algorithm>
  <public-key>base64encodedvalue==</public-key>
  <hidden-private-key/>
  <certificates>
    <certificate>
      <name>builtin-idevid-cert</name>
    </certificate>
    <certificate>
      <name>my-idevid-cert</name>
      <cert>base64encodedvalue==</cert>
    </certificate>
  </certificates>
</asymmetric-key>

<asymmetric-key>
  <name>encrypted-key</name>
  <algorithm>secp256r1</algorithm>
  <public-key>base64encodedvalue==</public-key>
  <encrypted-private-key>
    <symmetric-key-ref>operators-encrypted-key</symmetric-key-ref>
    <value>base64encodedvalue==</value>
  </encrypted-private-key>
</asymmetric-key>

<!-- Symmetric Keys -->
<symmetric-keys>
  <symmetric-key>
    <name>operators-encrypted-key</name>
    <algorithm>aes-256-cbc</algorithm>
    <encrypted-key>
      <symmetric-key-ref>tpm-protected-key</symmetric-key-ref>
      <value>base64encodedvalue==</value>
    </encrypted-key>
  </symmetric-key>
</symmetric-keys>
3.2.2. The "generate-symmetric-key" RPC

The following example illustrates the "generate-symmetric-key" RPC. The key being referenced is defined in the keystore example above.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <generate-symmetric-key
       xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
    <algorithm>aes-256-cbc</algorithm>
    <encrypt-with>
      <asymmetric-key-ref>tpm-protected-key</asymmetric-key-ref>
    </encrypt-with>
  </generate-symmetric-key>
</rpc>
```

Following is the complimentary RPC-reply.

```
<rpc-reply message-id="101"
           xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:ks="urn:ietf:params:xml:ns:yang:ietf-keystore">
  <ks:algorithm>aes-256-cbc</ks:algorithm>
  <ks:encrypted-key>
    <ks:asymmetric-key-ref>tpm-protected-key</ks:asymmetric-key-ref>
    <ks:value>base64encodedvalue==</ks:value>
  </ks:encrypted-key>
</rpc-reply>
```

3.2.3. Notable Keystore Groupings

The following non-normative module is used by subsequent examples to illustrate groupings defined in the ietf-crypto-types 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).

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

<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>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-usage>
<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>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>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)</name>
  <keystore-reference>
    <asymmetric-key>ex-rsa-key</asymmetric-key>
    <certificate>ex-rsa-cert</certificate>
  </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@2019-07-02.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

    This version of this YANG module is part of RFC XXXX

Watsen                   Expires January 3, 2020               [Page 18]
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 2019-07-02 {
  description
    "Initial version";
  reference
    "RFC VVVV: A YANG Data Model for a Keystore";
}

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

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

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

feature key-generation {
  description
    "Indicates that the server supports the actions related to the life cycling keys in <operational>. To be used by configuration, keys in <operational> must be copied to <running>.";
}

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

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

/* Groupings */

grouping key-reference-type-grouping {
  description
  "A reusable grouping for a choice for the type of key referenced in the keystore.";
  choice key-type {
    mandatory true;
    description
    "A choice between a reference to a symmetric or asymmetric key in the keystore.";
    leaf symmetric-key-ref {
      if-feature "keystore-supported";
      type leafref {
        path "/ks:keystore/ks:symmetric-keys/ks:symmetric-key/
          + "ks:name";
      }
      description
      "Identifies a symmetric key used to encrypt this key.";
    }
    leaf asymmetric-key-ref {
      if-feature "keystore-supported";
      type leafref {
        path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key/
          + "ks:name";
      }
      description
      "Identifies an asymmetric key used to encrypt this key.";
    }
  }
}

grouping encrypted-value-grouping {
  description
  "A reusable grouping for a value that has been encrypted by a symmetric or asymmetric key in the keystore.";
  uses "key-reference-type-grouping";
  leaf value {
    
  }
}
type binary;
description
"The private key, encrypted using the specified symmetric
or asymmetric key.";
}
}

grouping symmetric-key-grouping {
description
"This grouping is identical to the one in ietf-crypt-types
except that it adds a couple case statements enabling the
key value to be encrypted by a symmetric or an asymmetric
key known to the keystore.";
uses ct:symmetric-key-grouping {
augment "key-type" {
description
"Augments a new ‘case’ statement into the ‘choice’
statement defined by the ietf-crypto-types module.";
container encrypted-key {
description
"A container for the encrypted symmetric key value.";
uses encrypted-value-grouping;
}
}
}

 grouping asymmetric-key-pair-grouping {
description
"This grouping is identical to the one in ietf-crypt-types
except that it adds a couple case statements enabling the
key value to be encrypted by a symmetric or an asymmetric
key known to the keystore.";
uses ct:asymmetric-key-pair-grouping {
augment "private-key-type" {
description
"Augments a new ‘case’ statement into the ‘choice’
statement defined by the ietf-crypto-types module.";
container encrypted-private-key {
description
"A container for the encrypted asymmetric private
key value.";
uses encrypted-value-grouping;
}
}
}
grouping asymmetric-key-pair-with-cert-grouping {
  description
  "This grouping is identical to the one in ietf-crypt-types except that it adds a couple case statements enabling the key value to be encrypted by a symmetric or an asymmetric key known to the keystore."
  uses ct:asymmetric-key-pair-with-cert-grouping {
    augment "private-key-type" {
      description
      "Augments a new 'case' statement into the 'choice' statement defined by the ietf-crypto-types module."
      container encrypted-private-key {
        description
        "A container for the encrypted asymmetric private key value.";
        uses encrypted-value-grouping;
      }
    }
  }
}

grouping asymmetric-key-pair-with-certs-grouping {
  description
  "This grouping is identical to the one in ietf-crypt-types except that it adds a couple case statements enabling the key value to be encrypted by a symmetric or an asymmetric key known to the keystore."
  uses ct:asymmetric-key-pair-with-certs-grouping {
    augment "private-key-type" {
      description
      "Augments a new 'case' statement into the 'choice' statement defined by the ietf-crypto-types module."
      container encrypted-private-key {
        description
        "A container for the encrypted asymmetric private key value.";
        uses encrypted-value-grouping;
      }
    }
  }
}

grouping asymmetric-key-certificate-ref-grouping {
  leaf asymmetric-key {
    type ks:asymmetric-key-ref;
    must './..certificates';
    description
    "A reference to an asymmetric key in the keystore.";
  }
}
leaf certificate {
  type leafref {
    path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key[ks:" + "name = current()]/../asymmetric-key]/ks:certificates" + "/ks:certificate/ks:name";
  }
  must '../asymmetric-key';
  description "A reference to a specific certificate of the asymmetric key in the keystore.";
}

description "This grouping defines a reference to a specific certificate associated with an asymmetric key stored in the keystore."
}

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-definitions-supported";
      container local-definition {
        description "Container to hold the local key definition.";
        uses 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-definitions-supported";
            container local-definition {
                description
                "Container to hold the local key definition.";
                uses asymmetric-key-pair-with-certs-grouping;
            }
        }
        case keystore {
            if-feature "keystore-supported";
            leaf keystore-reference {
                type ks:asymmetric-key-ref;
                description
                "A reference to an asymmetric-key (and all of its
                associated certificates) 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-definitions-supported";
            container local-definition {
                description
                "Container to hold the local key definition.";
                uses asymmetric-key-pair-with-cert-grouping;
            }
        }
        case keystore {

if-feature "keystore-supported";
container keystore-reference {
  uses asymmetric-key-certificate-ref-grouping;
  description
  "A reference to a specific certificate (and its
   associated private key) in the keystore.";
}

description
"A choice between an inlined definition and a definition
that exists in the keystore.";
}

grouping keystore-grouping {
  description
  "Grouping definition enables use in other contexts. If ever
done, implementations SHOULD augment new 'case' statements
into local-or-keystore 'choice' statements to supply leafrefs
to the new location.";
  container asymmetric-keys {
    description
    "A list of asymmetric keys.";
    list asymmetric-key {
      key "name";
      description
      "An asymmetric key.";
      leaf name {
        type string;
        description
        "An arbitrary name for the asymmetric key.";
      }
      uses ks:asymmetric-key-pair-with-certs-grouping;
    }
  }
  container symmetric-keys {
    description
    "A list of symmetric keys.";
    list symmetric-key {
      key "name";
      description
      "A symmetric key.";
      leaf name {
        type string;
        description
        "An arbitrary name for the symmetric key.";
      }
      uses ks:symmetric-key-grouping;
    }
  }
}
container keystore {
    nacm:default-deny-write;
    description
        "The keystore contains a list of keys.";
    uses keystore-grouping;
}

rpc generate-symmetric-key {
    //nacm:default-deny-all;
    description
        "Requests the device to generate an symmetric key using
        the specified key algorithm, optionally encrypted using
        a key in the keystore. The output is this RPC can be
        used as input to a subsequent configuration request.";
    input {
        leaf algorithm {
            type ct:encryption-algorithm-t;
            mandatory true;
            description
                "The algorithm to be used when generating the key.";
            reference
                "RFC CCCC: Common YANG Data Types for Cryptography";
        }
        container encrypt-with {
            presence
                "Indicates that the key should be encrypted using
                the specified symmetric or asymmetric key. If not
                specified, then the private key is not encrypted
                when returned.";
            description
                "A container for the 'key-type' choice.";
            uses key-reference-type-grouping;
        }
    }
    output {
        uses ks:symmetric-key-grouping;
    }
} // end generate-symmetric-key
rpc generate-asymmetric-key {
  //nacm:default-deny-all;
  description
  "Requests the device to generate an asymmetric key using
  the specified key algorithm, optionally encrypted using
  a key in the keystore. The output is this RPC can be
  used as input to a subsequent configuration request."
  input {
    leaf algorithm {
      type ct:asymmetric-key-algorithm-t;
      mandatory true;
      description
      "The algorithm to be used when generating the key.";
      reference
      "RFC CCCC: Common YANG Data Types for Cryptography";
    }
    container encrypt-with {
      presence
      "Indicates that the key should be encrypted using
      the specified symmetric or asymmetric key. If not
      specified, then the private key is not encrypted
      when returned.";
      description
      "A container for the ‘key-type’ choice.";
      uses key-reference-type-grouping;
    }
  }
  output {
    uses ks:asymmetric-key-pair-grouping;
  }
} // end generate-asymmetric-key

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:

/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 than 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:

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 following registration is requested:

name: ietf-keystore
prefix: ks
reference: RFC VVVV

6. References

6.1. Normative References


6.2. Informative References


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 possibility 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>. 
A.10. 09 to 10

- Updated draft title to match new truststore draft title
- Moved everything under a top-level ‘grouping’ to enable use in other contexts.
- Renamed feature from ‘local-keys-supported’ to ‘local-definitions-supported’ (same name used in truststore)
- Removed the either-all-or-none ‘must’ expressions for the key’s 3-tuple values (since the values are now ‘mandatory true’ in crypto-types)
- Example updated to reflect ‘mandatory true’ change in crypto-types draft

A.11. 10 to 11

- Replaced typedef asymmetric-key-certificate-ref with grouping asymmetric-key-certificate-ref-grouping.
- Added feature feature ‘key-generation’.
- Cloned groupings symmetric-key-grouping, asymmetric-key-pair-grouping, asymmetric-key-pair-with-cert-grouping, and asymmetric-key-pair-with-certs-grouping from crypto-keys, augmenting into each new case statements for values that have been encrypted by other keys in the keystore. Refactored keystore model to use these groupings.
- Added new ‘symmetric-keys’ lists, as a sibling to the existing ‘asymmetric-keys’ list.
- Added RPCs (not actions) ‘generate-symmetric-key’ and ‘generate-asymmetric-key’ to *return* a (potentially encrypted) key.

A.12. 11 to 12

- Updated to reflect crypto-type’s draft using enumerations over identities.
Added examples for the 'generate-symmetric-key' and 'generate-asymmetric-key' RPCs.

Updated the Introduction section.

Acknowledgements

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

Author’s Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net