Workgroup: NETCONF Working Group
Internet-Draft: draft-ietf-netconf-keystore-20
Published: 20 August 2020
Intended Status: Standards Track
Expires: 21 February 2021
Authors: K. Watsen
Watsen Networks

# A YANG Data Model for a Keystore

## 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 or hidden. 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 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:

- AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
- CCCC --> 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:

- 2020-08-20 --> 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

## Copyright Notice

## Table of Contents

## 1. Introduction

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 or hidden (see [I-D.ietf-netconf-crypto-types]. 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 define 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 Platform Module (TPM). These systems are unique in that the cryptographic hardware hides the secret key values. Additionally, such hardware is commonly initiailized when manufactured to protect a "built-in" asymmetric key for which the public half is conveyed in an identity certificate (e.g., an IDevID [Std-802.1AR-2009] certificate). Please see Section 3 to see how built-in keys are supported.

This document intends to support existing practices; it does not intend to define new behvior for systems to implement. To simplify implementation, advanced key formats may be selectively implemented.

Implementations may utilize zero or more operating system level keystore utilities and/or hardware security modules (HSMs).

## 1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to define configuration modules for clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The normative dependency relationship between the various RFCs in the collection is presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Hyperlinks to each RFC are provided below the diagram.

```
                               crypto-types
                                 ^      ^
                                /        \
                               /          \
                      truststore         keystore
                       ^     ^             ^  ^
                       |     +---------+   |  |
                       |               |   |  |
                       |      +------------+  |
tcp-client-server      |     /         |      |
   ^    ^        ssh-client-server     |      |
   |    |           ^            tls-client-server
   |    |           |              ^     ^        http-client-server
   |    |           |              |     |                 ^
   |    |           |        +-----+     +---------+       |
   |    |           |        |                     |       |
   |    +-----------|--------|--------------+      |       |
   |                |        |              |      |       |
   +-----------+    |        |              |      |       |
               |    |        |              |      |       |
               |    |        |              |      |       |
            netconf-client-server       restconf-client-server
```

| Label in Diagram | Originating RFC |
|---|---|
| crypto-types | [I-D.ietf-netconf-crypto-types] |
| truststore | [I-D.ietf-netconf-trust-anchors] |
| keystore | [I-D.ietf-netconf-keystore] |

| | |
|---|---|
| tcp-client-server | [I-D.ietf-netconf-tcp-client-server] |
| ssh-client-server | [I-D.ietf-netconf-ssh-client-server] |
| tls-client-server | [I-D.ietf-netconf-tls-client-server] |
| http-client-server | [I-D.ietf-netconf-http-client-server] |
| netconf-client-server | [I-D.ietf-netconf-netconf-client-server] |
| restconf-client-server | [I-D.ietf-netconf-restconf-client-server] |

Table 1: Label to RFC Mapping

## 1.2. Specification 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.

## 1.3. Terminology

The terms "client" and "server" are defined in [RFC6241] and are not redefined here.

The term "keystore" is defined in this draft as a mechanism that intends safeguard secrets placed into it for protection.

The nomenclature "<running>" and "<operational>" are defined in [RFC8342].

The sentence fragments "augmented" and "augmented in" are used herein as the past tense verbified form of the "augment" statement defined in Section 7.17 of [RFC7950].

## 1.4. Adherence to the NMDA

This document is compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, keys and associated certificates installed during manufacturing (e.g., for an IDevID certificate) are expected to appear in <operational> (see Section 3).

## 2. The "ietf-keystore" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-keystore". A high-level overview of the module is provided in Section 2.1. Examples illustatrating the module's use are provided in Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

### 2.1. Data Model Overview

This section provides an overview of the "ietf-keystore" module in terms of its features, typedefs, groupings, and protocol-accessible nodes.

#### 2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-keystore" module:

```
Features:
  +-- keystore-supported
  +-- local-definitions-supported
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

#### 2.1.2. Typedefs

The following diagram lists the "typedef" statements defined in the "ietf-keystore" module:

```
Typedefs:
  leafref
    +-- symmetric-key-ref
    +-- asymmetric-key-ref
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

Comments:

* All of the typedefs defined in the "ietf-keystore" module extend the base "leafref" type defined in [RFC7950].

* The leafrefs refer to symmetric and asymmetric keys in the keystore. These typedefs are provided primarily as an aid to downstream modules that import the "ietf-keystore" module.

#### 2.1.3. Groupings

The following diagram lists all the "grouping" statements defined in the "ietf-keystore" module:

```
Groupings:
  +-- encrypted-by-choice-grouping
  +-- asymmetric-key-certificate-ref-grouping
  +-- local-or-keystore-symmetric-key-grouping
  +-- local-or-keystore-asymmetric-key-grouping
  +-- local-or-keystore-asymmetric-key-with-certs-grouping
  +-- local-or-keystore-end-entity-cert-with-key-grouping
  +-- keystore-grouping
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

Each of these groupings are presented in the following subsections.

#### 2.1.3.1. The "encrypted-by-choice-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "encrypted-by-choice-grouping" grouping:

The grouping's name is intended to be parsed "(encrypted-by)-(choice)-(grouping)", not as "(encrypted)-(by-choice)-(grouping)".

```
grouping encrypted-by-choice-grouping
  +-- (encrypted-by-choice)
     +--:(symmetric-key-ref)
     |  +-- symmetric-key-ref?    ks:symmetric-key-ref
     +--:(asymmetric-key-ref)
        +-- asymmetric-key-ref?   ks:asymmetric-key-ref
```

Comments:

* This grouping defines a "choice" statement with options to reference either a symmetric or an asymmetric key configured in the keystore.

#### 2.1.3.2. The "asymmetric-key-certificate-ref-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "asymmetric-key-certificate-ref-grouping" grouping:

```
grouping asymmetric-key-certificate-ref-grouping
  +-- asymmetric-key?   ks:asymmetric-key-ref
  +-- certificate?      leafref
```

Comments:

* This grouping defines a reference to a certificate in two parts: the first being the name of the asymmetric key the certificate is associated with, and the second being the name of the certificate itself.

#### 2.1.3.3. The "local-or-keystore-symmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-symmetric-key-grouping" grouping:

```
grouping local-or-keystore-symmetric-key-grouping
  +-- (local-or-keystore)
     +--:(local) {local-definitions-supported}?
     |  +-- local-definition
     |     +---u ct:symmetric-key-grouping
     +--:(keystore) {keystore-supported}?
        +-- keystore-reference?   ks:symmetric-key-ref
```

Comments:

*The "local-or-keystore-symmetric-key-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option for whether a symmetric key is defined locally or as a reference to a symmetric key in the keystore.

*A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a symmetric key in an alternate location.

*For the "local-definition" option, the defintion uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [I-D.ietf-netconf-crypto-types].

*For the "keystore" option, the "keystore-reference" is an instance of the "symmetric-key-ref" discussed in Section 2.1.2.

#### 2.1.3.4. The "local-or-keystore-asymmetric-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-asymmetric-key-grouping" grouping:

```
grouping local-or-keystore-asymmetric-key-grouping
  +-- (local-or-keystore)
     +--:(local) {local-definitions-supported}?
     |  +-- local-definition
     |     +---u ct:asymmetric-key-pair-grouping
     +--:(keystore) {keystore-supported}?
        +-- keystore-reference?   ks:asymmetric-key-ref
```

Comments:

*The "local-or-keystore-asymmetric-key-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option for whether an asymmetric key is defined locally or as a reference to an asymmetric key in the keystore.

* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference an asymmetric key in an alternate location.

* For the "local-definition" option, the defintion uses the "asymmetric-key-pair-grouping" grouping discussed in Section 2.1.4.5 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

### 2.1.3.5. The "local-or-keystore-asymmetric-key-with-certs-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-asymmetric-key-with-certs-grouping" grouping:

```
grouping local-or-keystore-asymmetric-key-with-certs-grouping
  +-- (local-or-keystore)
     +--:(local) {local-definitions-supported}?
     |  +-- local-definition
     |     +---u ct:asymmetric-key-pair-with-certs-grouping
     +--:(keystore) {keystore-supported}?
        +-- keystore-reference?   ks:asymmetric-key-ref
```

Comments:

* The "local-or-keystore-asymmetric-key-with-certs-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option for whether an asymmetric key is defined locally or as a reference to an asymmetric key in the keystore.

* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference an asymmetric key in an alternate location.

* For the "local-definition" option, the defintion uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

#### 2.1.3.6. The "local-or-keystore-end-entity-cert-with-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "local-or-keystore-end-entity-cert-with-key-grouping" grouping:

```
grouping local-or-keystore-end-entity-cert-with-key-grouping
  +-- (local-or-keystore)
     +--:(local) {local-definitions-supported}?
     |  +-- local-definition
     |     +---u ct:asymmetric-key-pair-with-cert-grouping
     +--:(keystore) {keystore-supported}?
        +-- keystore-reference
           +---u asymmetric-key-certificate-ref-grouping
```

Comments:

* The "local-or-keystore-end-entity-cert-with-key-grouping" grouping is provided soley as convenience to downstream modules that wish to offer an option for whether a symmetric key is defined locally or as a reference to a symmetric key in the keystore.

* A "choice" statement is used to expose the various options. Each option is enabled by a "feature" statement. Additional "case" statements MAY be augmented in if, e.g., there is a need to reference a symmetric key in an alternate location.

* For the "local-definition" option, the defintion uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 of [I-D.ietf-netconf-crypto-types].

* For the "keystore" option, the "keystore-reference" uses the "asymmetric-key-certificate-ref-grouping" grouping discussed in Section 2.1.3.2.

#### 2.1.3.7. The "keystore-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "keystore-grouping" grouping:

```
grouping keystore-grouping
  +-- asymmetric-keys
  |  +-- asymmetric-key* [name]
  |     +-- name?                                    string
  |     +---u ct:asymmetric-key-pair-with-certs-grouping
  +-- symmetric-keys
     +-- symmetric-key* [name]
        +-- name?                        string
        +---u ct:symmetric-key-grouping
```

Comments:

* The "keystore-grouping" grouping defines a keystore instance as being composed of symmetric and asymmetric keys. The stucture for the symmetric and asymmetric keys is essentially the same, being a "list" inside a "container".
* For asymmetric keys, each "asymmetric-key" uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.11 of [I-D.ietf-netconf-crypto-types].
* For symmetric keys, each "symmetric-key" uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [I-D.ietf-netconf-crypto-types].

### 2.1.4. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-keystore" module, without expanding the "grouping" statements:

```
module: ietf-keystore
  +--rw keystore
     +---u keystore-grouping

  grouping encrypted-by-choice-grouping
    +-- (encrypted-by-choice)
       +--:(symmetric-key-ref)
       |  +-- symmetric-key-ref?    ks:symmetric-key-ref
       +--:(asymmetric-key-ref)
          +-- asymmetric-key-ref?   ks:asymmetric-key-ref
  grouping asymmetric-key-certificate-ref-grouping
    +-- asymmetric-key?   ks:asymmetric-key-ref
    +-- certificate?      leafref
  grouping local-or-keystore-symmetric-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +---u ct:symmetric-key-grouping
       +--:(keystore) {keystore-supported}?
          +-- keystore-reference?   ks:symmetric-key-ref
  grouping local-or-keystore-asymmetric-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +---u ct:asymmetric-key-pair-grouping
       +--:(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
       |     +---u ct:asymmetric-key-pair-with-certs-grouping
       +--:(keystore) {keystore-supported}?
          +-- keystore-reference?   ks:asymmetric-key-ref
  grouping local-or-keystore-end-entity-cert-with-key-grouping
    +-- (local-or-keystore)
       +--:(local) {local-definitions-supported}?
       |  +-- local-definition
       |     +---u ct:asymmetric-key-pair-with-cert-grouping
       +--:(keystore) {keystore-supported}?
          +-- keystore-reference
             +---u asymmetric-key-certificate-ref-grouping
  grouping keystore-grouping
    +-- asymmetric-keys
    |  +-- asymmetric-key* [name]
    |     +-- name?                                         string
    |     +---u ct:asymmetric-key-pair-with-certs-grouping
    +-- symmetric-keys
       +-- symmetric-key* [name]
          +-- name?                        string
          +---u ct:symmetric-key-grouping
```

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-keystore" module, with all "grouping" statements expanded, enabling the keystore's full structure to be seen:

```
module: ietf-keystore
  +--rw keystore
     +--rw asymmetric-keys
     |  +--rw asymmetric-key* [name]
     |     +--rw name                                  string
     |     +--rw public-key-format                     identityref
     |     +--rw public-key                            binary
     |     +--rw private-key-format?                   identityref
     |     +--rw (private-key-type)
     |     |  +--:(cleartext-private-key)
     |     |  |  +--rw cleartext-private-key?            binary
     |     |  +--:(hidden-private-key)
     |     |  |  +--rw hidden-private-key?               empty
     |     |  +--:(encrypted-private-key) {private-key-encryption}?
     |     |     +--rw encrypted-private-key
     |     |        +--rw encrypted-by
     |     |        |  +--rw (encrypted-by-choice)
     |     |        |     +--:(symmetric-key-ref)
     |     |        |     |  +--rw symmetric-key-ref?
     |     |        |     |          ks:symmetric-key-ref
     |     |        |     +--:(asymmetric-key-ref)
     |     |        |        +--rw asymmetric-key-ref?
     |     |        |                ks:asymmetric-key-ref
     |     |        +--rw encrypted-value    binary
     |     +--rw certificates
     |     |  +--rw certificate* [name]
     |     |     +--rw name                       string
     |     |     +--rw cert-data                  end-entity-cert-cms
     |     |     +---n certificate-expiration
     |     |             {certificate-expiration-notification}?
     |     |        +-- expiration-date    yang:date-and-time
     |     +---x generate-certificate-signing-request
     |             {certificate-signing-request-generation}?
     |        +---w input
     |        |  +---w csr-info    ct:csr-info
     |        +--ro output
     |           +--ro certificate-signing-request    ct:csr
     +--rw symmetric-keys
        +--rw symmetric-key* [name]
           +--rw name                  string
           +--rw key-format?           identityref
           +--rw (key-type)
              +--:(cleartext-key)
              |  +--rw cleartext-key?   binary
              +--:(hidden-key)
              |  +--rw hidden-key?      empty
              +--:(encrypted-key) {symmetric-key-encryption}?
                 +--rw encrypted-key
                    +--rw encrypted-by
```

```
                     |  +--rw (encrypted-by-choice)
                     |     +--:(symmetric-key-ref)
                     |     |  +--rw symmetric-key-ref?
                     |     |          ks:symmetric-key-ref
                     |     +--:(asymmetric-key-ref)
                     |        +--rw asymmetric-key-ref?
                     |                ks:asymmetric-key-ref
                     +--rw encrypted-value    binary

   grouping encrypted-by-choice-grouping
     +-- (encrypted-by-choice)
        +--:(symmetric-key-ref)
        |  +-- symmetric-key-ref?    ks:symmetric-key-ref
        +--:(asymmetric-key-ref)
           +-- asymmetric-key-ref?   ks:asymmetric-key-ref
   grouping asymmetric-key-certificate-ref-grouping
     +-- asymmetric-key?   ks:asymmetric-key-ref
     +-- certificate?      leafref
   grouping local-or-keystore-symmetric-key-grouping
     +-- (local-or-keystore)
        +--:(local) {local-definitions-supported}?
        |  +-- local-definition
        |     +-- key-format?            identityref
        |     +-- (key-type)
        |        +--:(cleartext-key)
        |        |  +-- cleartext-key?   binary
        |        +--:(hidden-key)
        |        |  +-- hidden-key?      empty
        |        +--:(encrypted-key) {symmetric-key-encryption}?
        |           +-- encrypted-key
        |              +-- encrypted-by
        |              +-- encrypted-value    binary
        +--:(keystore) {keystore-supported}?
           +-- keystore-reference?   ks:symmetric-key-ref
   grouping local-or-keystore-asymmetric-key-grouping
     +-- (local-or-keystore)
        +--:(local) {local-definitions-supported}?
        |  +-- local-definition
        |     +-- public-key-format              identityref
        |     +-- public-key                     binary
        |     +-- private-key-format?            identityref
        |     +-- (private-key-type)
        |        +--:(cleartext-private-key)
        |        |  +-- cleartext-private-key?   binary
        |        +--:(hidden-private-key)
        |        |  +-- hidden-private-key?      empty
        |        +--:(encrypted-private-key) {private-key-encryption}?
        |           +-- encrypted-private-key
        |              +-- encrypted-by
     |              +-- encrypted-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
     |     +-- public-key-format                        identityref
     |     +-- public-key                               binary
     |     +-- private-key-format?                      identityref
     |     +-- (private-key-type)
     |     |  +--:(cleartext-private-key)
     |     |  |  +-- cleartext-private-key?            binary
     |     |  +--:(hidden-private-key)
     |     |  |  +-- hidden-private-key?               empty
     |     |  +--:(encrypted-private-key) {private-key-encryption}?
     |     |     +-- encrypted-private-key
     |     |        +-- encrypted-by
     |     |        +-- encrypted-value    binary
     |     +-- certificates
     |     |  +-- certificate* [name]
     |     |     +-- name?                     string
     |     |     +-- cert-data                 end-entity-cert-cms
     |     |     +---n certificate-expiration
     |     |             {certificate-expiration-notification}?
     |     |        +-- expiration-date    yang:date-and-time
     |     +---x generate-certificate-signing-request
     |             {certificate-signing-request-generation}?
     |        +---w input
     |        |  +---w csr-info    ct:csr-info
     |        +--ro output
     |           +--ro certificate-signing-request    ct:csr
     +--:(keystore) {keystore-supported}?
        +-- keystore-reference?   ks:asymmetric-key-ref
grouping local-or-keystore-end-entity-cert-with-key-grouping
  +-- (local-or-keystore)
     +--:(local) {local-definitions-supported}?
     |  +-- local-definition
     |     +-- public-key-format                        identityref
     |     +-- public-key                               binary
     |     +-- private-key-format?                      identityref
     |     +-- (private-key-type)
     |     |  +--:(cleartext-private-key)
     |     |  |  +-- cleartext-private-key?            binary
     |     |  +--:(hidden-private-key)
     |     |  |  +-- hidden-private-key?               empty
     |     |  +--:(encrypted-private-key) {private-key-encryption}?
     |     |     +-- encrypted-private-key
     |     |        +-- encrypted-by
```

```
       |     |        +-- encrypted-value    binary
       |     +-- cert-data?
       |     |       end-entity-cert-cms
       |     +---n certificate-expiration
       |     |       {certificate-expiration-notification}?
       |     |  +-- expiration-date    yang:date-and-time
       |     +---x generate-certificate-signing-request
       |             {certificate-signing-request-generation}?
       |        +---w input
       |        |  +---w csr-info    ct:csr-info
       |        +--ro output
       |           +--ro certificate-signing-request    ct:csr
       +--:(keystore) {keystore-supported}?
          +-- keystore-reference
             +-- asymmetric-key?   ks:asymmetric-key-ref
             +-- certificate?      leafref
  grouping keystore-grouping
    +-- asymmetric-keys
    |  +-- asymmetric-key* [name]
    |     +-- name?                                   string
    |     +-- public-key-format                       identityref
    |     +-- public-key                              binary
    |     +-- private-key-format?                     identityref
    |     +-- (private-key-type)
    |     |  +--:(cleartext-private-key)
    |     |  |  +-- cleartext-private-key?            binary
    |     |  +--:(hidden-private-key)
    |     |  |  +-- hidden-private-key?               empty
    |     |  +--:(encrypted-private-key) {private-key-encryption}?
    |     |     +-- encrypted-private-key
    |     |        +-- encrypted-by
    |     |        +-- encrypted-value    binary
    |     +-- certificates
    |     |  +-- certificate* [name]
    |     |     +-- name?                     string
    |     |     +-- cert-data                 end-entity-cert-cms
    |     |     +---n certificate-expiration
    |     |              {certificate-expiration-notification}?
    |     |        +-- expiration-date    yang:date-and-time
    |     +---x generate-certificate-signing-request
    |             {certificate-signing-request-generation}?
    |        +---w input
    |        |  +---w csr-info    ct:csr-info
    |        +--ro output
    |           +--ro certificate-signing-request    ct:csr
    +-- symmetric-keys
       +-- symmetric-key* [name]
          +-- name?                  string
          +-- key-format?            identityref
```

```
+-- (key-type)
   +--:(cleartext-key)
   |  +-- cleartext-key?   binary
   +--:(hidden-key)
   |  +-- hidden-key?      empty
   +--:(encrypted-key) {symmetric-key-encryption}?
      +-- encrypted-key
         +-- encrypted-by
         +-- encrypted-value    binary
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].
* The protcol-accessible nodes for the "ietf-keystore" module are an instance of the "keystore-grouping" grouping discussed in Section 2.1.3.7.
* The reason for why "keystore-grouping" exists separate from the protocol-accessible nodes definition is so as to enable instances of the keystore to be instantiated in other locations, as may be needed or desired by some modules.

## 2.2. Example Usage

The examples in this section are encoded using XML, such as might be the case when using the NETCONF protocol. Other encodings MAY be used, such as JSON when using the RESTCONF protocol.

### 2.2.1. A Keystore Instance

The following example illustrates keys in <running>. Please see Section 3 for an example illustrating built-in values in <operational>.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
   xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

   <symmetric-keys>
      <symmetric-key>
         <name>cleartext-symmetric-key</name>
         <key-format>ct:octet-string-key-format</key-format>
         <cleartext-key>base64encodedvalue==</cleartext-key>
      </symmetric-key>
      <symmetric-key>
         <name>hidden-symmetric-key</name>
         <hidden-key/>
      </symmetric-key>
      <symmetric-key>
         <name>encrypted-symmetric-key</name>
         <key-format>
            ct:encrypted-one-symmetric-key-format
         </key-format>
         <encrypted-key>
           <encrypted-by>
             <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-k\
ey-ref>
           </encrypted-by>
           <encrypted-value>base64encodedvalue==</encrypted-value>
         </encrypted-key>
      </symmetric-key>
   </symmetric-keys>

   <asymmetric-keys>
      <asymmetric-key>
         <name>ssh-rsa-key</name>
         <public-key-format>
            ct:ssh-public-key-format
         </public-key-format>
         <public-key>base64encodedvalue==</public-key>
         <private-key-format>
            ct:rsa-private-key-format
         </private-key-format>
         <cleartext-private-key>base64encodedvalue==</cleartext-priv\
ate-key>
      </asymmetric-key>
      <asymmetric-key>
         <name>ssh-rsa-key-with-cert</name>
         <public-key-format>
            ct:subject-public-key-info-format
         </public-key-format>
         <public-key>base64encodedvalue==</public-key>
         <private-key-format>
            ct:rsa-private-key-format
         </private-key-format>
         <cleartext-private-key>base64encodedvalue==</cleartext-priv\
ate-key>
         <certificates>
            <certificate>
               <name>ex-rsa-cert2</name>
               <cert-data>base64encodedvalue==</cert-data>
            </certificate>
         </certificates>
      </asymmetric-key>
      <asymmetric-key>
         <name>raw-private-key</name>
         <public-key-format>
            ct:subject-public-key-info-format
         </public-key-format>
         <public-key>base64encodedvalue==</public-key>
         <private-key-format>
            ct:rsa-private-key-format
         </private-key-format>
         <cleartext-private-key>base64encodedvalue==</cleartext-priv\
ate-key>
      </asymmetric-key>
      <asymmetric-key>
         <name>rsa-asymmetric-key</name>
         <public-key-format>
            ct:subject-public-key-info-format
         </public-key-format>
         <public-key>base64encodedvalue==</public-key>
         <private-key-format>
            ct:rsa-private-key-format
         </private-key-format>
         <cleartext-private-key>base64encodedvalue==</cleartext-priv\
ate-key>
         <certificates>
            <certificate>
               <name>ex-rsa-cert</name>
               <cert-data>base64encodedvalue==</cert-data>
            </certificate>
         </certificates>
      </asymmetric-key>
      <asymmetric-key>
         <name>ec-asymmetric-key</name>
         <public-key-format>
            ct:subject-public-key-info-format
         </public-key-format>
         <public-key>base64encodedvalue==</public-key>
         <private-key-format>
```

```
            ct:ec-private-key-format
         </private-key-format>
         <cleartext-private-key>base64encodedvalue==</cleartext-priv\
ate-key>
         <certificates>
            <certificate>
               <name>ex-ec-cert</name>
               <cert-data>base64encodedvalue==</cert-data>
            </certificate>
         </certificates>
      </asymmetric-key>
      <asymmetric-key>
         <name>hidden-asymmetric-key</name>
         <public-key-format>
            ct:subject-public-key-info-format
         </public-key-format>
         <public-key>base64encodedvalue==</public-key>
         <hidden-private-key/>
         <certificates>
            <certificate>
               <name>builtin-idevid-cert</name>
               <cert-data>base64encodedvalue==</cert-data>
            </certificate>
            <certificate>
               <name>my-ldevid-cert</name>
               <cert-data>base64encodedvalue==</cert-data>
            </certificate>
         </certificates>
      </asymmetric-key>
      <asymmetric-key>
         <name>encrypted-asymmetric-key</name>
         <public-key-format>
            ct:subject-public-key-info-format
         </public-key-format>
         <public-key>base64encodedvalue==</public-key>
         <private-key-format>
            ct:encrypted-one-asymmetric-key-format
         </private-key-format>
         <encrypted-private-key>
           <encrypted-by>
             <symmetric-key-ref>encrypted-symmetric-key</symmetric-k\
ey-ref>
           </encrypted-by>
           <encrypted-value>base64encodedvalue==</encrypted-value>
         </encrypted-private-key>
      </asymmetric-key>
   </asymmetric-keys>
</keystore>
```

### 2.2.2. A Certificate Expiration Notification

The following example illustrates a "certificate-expiration" notification for a certificate associated with a key configured in the keystore.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
    <asymmetric-keys>
      <asymmetric-key>
        <name>hidden-asymmetric-key</name>
        <certificates>
          <certificate>
            <name>my-ldevid-cert</name>
            <certificate-expiration>
              <expiration-date>2018-08-05T14:18:53-05:00</expiration\
-date>
            </certificate-expiration>
          </certificate>
        </certificates>
      </asymmetric-key>
    </asymmetric-keys>
  </keystore>
</notification>
```

### 2.2.3. The "Local or Keystore" Groupings

This section illustrates the various "local-or-keystore" groupings defined in the "ietf-keystore" module, specifically the "local-or-keystore-symmetric-key-grouping" (Section 2.1.3.3), "local-or-keystore-asymmetric-key-grouping" (Section 2.1.3.4), "local-or-keystore-asymmetric-key-with-certs-grouping" (Section 2.1.3.5), and "local-or-keystore-end-entity-cert-with-key-grouping" (Section 2.1.3.6) groupings.

These examples assume the existence of an example module called "ex-keystore-usage" having the namespace "http://example.com/ns/example-keystore-usage".

The ex-keystore-usage module is first presented using tree diagrams [RFC8340], followed by an instance example illustrating all the "local-or-keystore" groupings in use, followed by the YANG module itself.

The following tree diagram illustrates "ex-keystore-usage" without
expanding the "grouping" statements:

```
module: ex-keystore-usage
  +--rw keystore-usage
     +--rw symmetric-key* [name]
     |  +--rw name                                         string
     |  +---u ks:local-or-keystore-symmetric-key-grouping
     +--rw asymmetric-key* [name]
     |  +--rw name                                          string
     |  +---u ks:local-or-keystore-asymmetric-key-grouping
     +--rw asymmetric-key-with-certs* [name]
     |  +--rw name
     |  |       string
     |  +---u ks:local-or-keystore-asymmetric-key-with-certs-grouping
     +--rw end-entity-cert-with-key* [name]
        +--rw name
        |       string
        +---u ks:local-or-keystore-end-entity-cert-with-key-grouping
```

The following tree diagram illustrates the "ex-keystore-usage"
module, with all "grouping" statements expanded, enabling the
keystore's full structure to be seen:

```
module: ex-keystore-usage
  +--rw keystore-usage
     +--rw symmetric-key* [name]
     |  +--rw name                        string
     |  +--rw (local-or-keystore)
     |     +--:(local) {local-definitions-supported}?
     |     |  +--rw local-definition
     |     |     +--rw key-format?           identityref
     |     |     +--rw (key-type)
     |     |        +--:(cleartext-key)
     |     |        |  +--rw cleartext-key?   binary
     |     |        +--:(hidden-key)
     |     |        |  +--rw hidden-key?      empty
     |     |        +--:(encrypted-key) {symmetric-key-encryption}?
     |     |           +--rw encrypted-key
     |     |              +--rw encrypted-by
     |     |              +--rw encrypted-value    binary
     |     +--:(keystore) {keystore-supported}?
     |        +--rw keystore-reference?   ks:symmetric-key-ref
     +--rw asymmetric-key* [name]
     |  +--rw name                        string
     |  +--rw (local-or-keystore)
     |     +--:(local) {local-definitions-supported}?
     |     |  +--rw local-definition
     |     |     +--rw public-key-format              identityref
     |     |     +--rw public-key                     binary
     |     |     +--rw private-key-format?            identityref
     |     |     +--rw (private-key-type)
     |     |        +--:(cleartext-private-key)
     |     |        |  +--rw cleartext-private-key?   binary
     |     |        +--:(hidden-private-key)
     |     |        |  +--rw hidden-private-key?      empty
     |     |        +--:(encrypted-private-key)
     |     |                 {private-key-encryption}?
     |     |           +--rw encrypted-private-key
     |     |              +--rw encrypted-by
     |     |              +--rw encrypted-value    binary
     |     +--:(keystore) {keystore-supported}?
     |        +--rw keystore-reference?   ks:asymmetric-key-ref
     +--rw asymmetric-key-with-certs* [name]
     |  +--rw name                        string
     |  +--rw (local-or-keystore)
     |     +--:(local) {local-definitions-supported}?
     |     |  +--rw local-definition
     |     |     +--rw public-key-format
     |     |     |       identityref
     |     |     +--rw public-key                              binary
     |     |     +--rw private-key-format?
     |     |     |       identityref
     |     |     +--rw (private-key-type)
     |     |     |  +--:(cleartext-private-key)
     |     |     |  |  +--rw cleartext-private-key?            binary
     |     |     |  +--:(hidden-private-key)
     |     |     |  |  +--rw hidden-private-key?               empty
     |     |     |  +--:(encrypted-private-key)
     |     |     |           {private-key-encryption}?
     |     |     |     +--rw encrypted-private-key
     |     |     |        +--rw encrypted-by
     |     |     |        +--rw encrypted-value    binary
     |     |     +--rw certificates
     |     |     |  +--rw certificate* [name]
     |     |     |     +--rw name                      string
     |     |     |     +--rw cert-data
     |     |     |     |       end-entity-cert-cms
     |     |     |     +---n certificate-expiration
     |     |     |             {certificate-expiration-notification}?
     |     |     |        +-- expiration-date    yang:date-and-time
     |     |     +---x generate-certificate-signing-request
     |     |             {certificate-signing-request-generation}?
     |     |        +---w input
     |     |        |  +---w csr-info    ct:csr-info
     |     |        +--ro output
     |     |           +--ro certificate-signing-request    ct:csr
     |     +--:(keystore) {keystore-supported}?
     |        +--rw keystore-reference?   ks:asymmetric-key-ref
     +--rw end-entity-cert-with-key* [name]
        +--rw name                        string
        +--rw (local-or-keystore)
           +--:(local) {local-definitions-supported}?
           |  +--rw local-definition
           |     +--rw public-key-format
           |     |       identityref
           |     +--rw public-key                              binary
           |     +--rw private-key-format?
           |     |       identityref
           |     +--rw (private-key-type)
           |     |  +--:(cleartext-private-key)
           |     |  |  +--rw cleartext-private-key?            binary
           |     |  +--:(hidden-private-key)
           |     |  |  +--rw hidden-private-key?               empty
           |     |  +--:(encrypted-private-key)
           |     |           {private-key-encryption}?
           |     |     +--rw encrypted-private-key
           |     |        +--rw encrypted-by
           |     |        +--rw encrypted-value    binary
           |     +--rw cert-data?
           |     |       end-entity-cert-cms
           |     +---n certificate-expiration
     |     |       {certificate-expiration-notification}?
     |     |  +-- expiration-date    yang:date-and-time
     |     +---x generate-certificate-signing-request
     |             {certificate-signing-request-generation}?
     |        +---w input
     |        |  +---w csr-info    ct:csr-info
     |        +--ro output
     |           +--ro certificate-signing-request    ct:csr
     +--:(keystore) {keystore-supported}?
        +--rw keystore-reference
           +--rw asymmetric-key?   ks:asymmetric-key-ref
           +--rw certificate?      leafref
```

The following example provides two equivalent instances of each grouping, the first being a reference to a keystore and the second being locally-defined. The instance having a reference to a keystore is consistent with the keystore defined in Section 2.2.1. The two instances are equivalent, as the locally-defined instance example contains the same values defined by the keystore instance referenced by its sibling example.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

<keystore-usage
  xmlns="http://example.com/ns/example-keystore-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

  <!-- The following two equivalent examples illustrate the -->
  <!-- "local-or-keystore-symmetric-key-grouping" grouping: -->

  <symmetric-key>
    <name>example 1a</name>
    <keystore-reference>cleartext-symmetric-key</keystore-reference>
  </symmetric-key>

  <symmetric-key>
    <name>example 1b</name>
    <local-definition>
      <key-format>ct:octet-string-key-format</key-format>
      <cleartext-key>base64encodedvalue==</cleartext-key>
    </local-definition>
  </symmetric-key>


  <!-- The following two equivalent examples illustrate the  -->
  <!-- "local-or-keystore-asymmetric-key-grouping" grouping: -->

  <asymmetric-key>
    <name>example 2a</name>
    <keystore-reference>rsa-asymmetric-key</keystore-reference>
  </asymmetric-key>

  <asymmetric-key>
    <name>example 2b</name>
    <local-definition>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <private-key-format>
        ct:rsa-private-key-format
      </private-key-format>
      <cleartext-private-key>base64encodedvalue==</cleartext-private\
-key>
    </local-definition>
  </asymmetric-key>


  <!-- the following two equivalent examples illustrate        -->
  <!-- "local-or-keystore-asymmetric-key-with-certs-grouping": -->
```

```
   <asymmetric-key-with-certs>
     <name>example 3a</name>
     <keystore-reference>rsa-asymmetric-key</keystore-reference>
   </asymmetric-key-with-certs>

   <asymmetric-key-with-certs>
     <name>example 3b</name>
     <local-definition>
       <public-key-format>
           ct:subject-public-key-info-format
       </public-key-format>
       <public-key>base64encodedvalue==</public-key>
       <private-key-format>
         ct:rsa-private-key-format
       </private-key-format>
       <cleartext-private-key>base64encodedvalue==</cleartext-private\
 -key>
       <certificates>
         <certificate>
           <name>a locally-defined cert</name>
           <cert-data>base64encodedvalue==</cert-data>
         </certificate>
       </certificates>
     </local-definition>
   </asymmetric-key-with-certs>


   <!-- The following two equivalent examples illustrate       -->
   <!-- "local-or-keystore-end-entity-cert-with-key-grouping": -->

   <end-entity-cert-with-key>
     <name>example 4a</name>
     <keystore-reference>
       <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
       <certificate>ex-rsa-cert</certificate>
     </keystore-reference>
   </end-entity-cert-with-key>

   <end-entity-cert-with-key>
     <name>example 4b</name>
     <local-definition>
       <public-key-format>
         ct:subject-public-key-info-format
       </public-key-format>
       <public-key>base64encodedvalue==</public-key>
       <private-key-format>
         ct:rsa-private-key-format
       </private-key-format>
       <cleartext-private-key>base64encodedvalue==</cleartext-private\
-key>
      <cert-data>base64encodedvalue==</cert-data>
    </local-definition>
  </end-entity-cert-with-key>

</keystore-usage>
```

Following is the "ex-keystore-usage" module's YANG definition:

```
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 CCCC: A YANG Data Model for a Keystore";
  }

  organization
   "Example Corporation";

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

  description
   "This module illustrates notable groupings defined in
    the 'ietf-keystore' module.";

  revision "2020-08-20" {
    description
     "Initial version";
    reference
     "RFC CCCC: A YANG Data Model for a Keystore";
  }

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

    list symmetric-key {
      key name;
      leaf name {
        type string;
        description
          "An arbitrary name for this key.";
      }
      uses ks:local-or-keystore-symmetric-key-grouping;
      description
        "An symmetric key that may be configured locally or be a
         reference to a symmetric key in the keystore.";
    }

    list asymmetric-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 asymmetric-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 asymmetric
           key, that may be configured locally or be a reference
           to another certificate (and its associated asymmetric
           key) in the keystore.";
      }
    }

  }
```

## 2.3.  YANG Module

This YANG module has normative references to [RFC8341] and [I-D.ietf-netconf-crypto-types].

```
<CODE BEGINS> file "ietf-keystore@2020-08-20.yang"
```

```
module ietf-keystore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-keystore";
  prefix ks;

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

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

  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) 2020 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 CCCC
     (https://www.rfc-editor.org/info/rfcCCCC); 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 2020-08-20 {
  description
    "Initial version";
  reference
    "RFC CCCC: 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.";
}

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

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

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.";
}

/*****************/
/*   Groupings   */
```

```
/*****************/

grouping encrypted-by-choice-grouping {
  description
    "A grouping that defines a choice that can be augmented
     into the 'encrypted-by' node presented by the
     'symmetric-key-grouping' and 'asymmetric-key-pair-grouping'
     groupings defined in RFC AAAA.";
  choice encrypted-by-choice {
    nacm:default-deny-write;
    mandatory true;
    description
      "A choice amongst other symmetric or asymmetric keys.";
    case symmetric-key-ref {
      leaf symmetric-key-ref {
        type ks:symmetric-key-ref;
        description
         "Identifies the symmetric key used to encrypt the
          associated key.";
      }
    }
    case asymmetric-key-ref {
      leaf asymmetric-key-ref {
        type ks:asymmetric-key-ref;
        description
         "Identifies the asymmetric key whose public key
          encrypted the associated key.";
      }
    }
  }
}

grouping asymmetric-key-certificate-ref-grouping {
  description
    "This grouping defines a reference to a specific certificate
     associated with an asymmetric key stored in the Keystore.";
  leaf asymmetric-key {
    nacm:default-deny-write;
    type ks:asymmetric-key-ref;
    must '../certificate';
    description
      "A reference to an asymmetric key in the Keystore.";
  }
  leaf certificate {
    nacm:default-deny-write;
    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.";
  }
}


// local-or-keystore-* groupings

grouping local-or-keystore-symmetric-key-grouping {
  description
    "A grouping that expands to allow the symmetric key to be
     either stored locally, i.e., within the using data model,
     or a reference to a symmetric key stored in the Keystore.";
  choice local-or-keystore {
    nacm:default-deny-write;
    mandatory true;
    description
      "A choice between an inlined definition and a definition
       that exists in the Keystore.";
    case local {
      if-feature "local-definitions-supported";
      container local-definition {
        description
          "Container to hold the local key definition.";
        uses ct:symmetric-key-grouping;
      }
    }
    case keystore {
      if-feature "keystore-supported";
      leaf keystore-reference {
        type ks:symmetric-key-ref;
        description
          "A reference to an symmetric key that exists 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, i.e., within the using data model,
     or a reference to an asymmetric key stored in the Keystore.";
  choice local-or-keystore {
    nacm:default-deny-write;
    mandatory true;
    description
      "A choice between an inlined definition and a definition
       that exists in the Keystore.";
    case local {
      if-feature "local-definitions-supported";
      container local-definition {
        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 without any regard for any certificates
           that may be associated with it.";
      }
    }
  }
}

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,
     i.e., within the using data model, or a reference to an
     asymmetric key (and its associated certificates) stored
     in the Keystore.";
  choice local-or-keystore {
    nacm:default-deny-write;
    mandatory true;
    description
      "A choice between an inlined definition and a definition
       that exists in the Keystore.";
    case local {
      if-feature "local-definitions-supported";
      container local-definition {
        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 an asymmetric-key (and all of its
           associated certificates) 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 asymmetric key pair) to be either stored
     locally, i.e., within the using data model, or a reference
     to a specific certificate in the Keystore.";
  choice local-or-keystore {
    nacm:default-deny-write;
    mandatory true;
    description
      "A choice between an inlined definition and a definition
       that exists in the Keystore.";
    case local {
      if-feature "local-definitions-supported";
      container local-definition {
        description
          "Container to hold the local key definition.";
        uses ct: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 associated with
           an asymmetric key stored 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 {
    nacm:default-deny-write;
    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 ct:asymmetric-key-pair-with-certs-grouping;
    }
  }
  container symmetric-keys {
    nacm:default-deny-write;
    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 ct:symmetric-key-grouping;
    }
  }
} // grouping keystore-grouping



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

container keystore {
  description
    "The Keystore contains a list of symmetric keys and a list
     of asymmetric keys.";
  nacm:default-deny-write;
  uses keystore-grouping {
    augment "symmetric-keys/symmetric-key/key-type/encrypted-key/"
            + "encrypted-key/encrypted-by" {
      description
        "Augments in a choice statement enabling the encrypting
```

```
             key to be any other symmetric or asymmetric key in the
             keystore.";
          uses encrypted-by-choice-grouping;
        }
        augment "asymmetric-keys/asymmetric-key/private-key-type/"
                + "encrypted-private-key/encrypted-private-key/"
                + "encrypted-by" {
          description
            "Augments in a choice statement enabling the encrypting
             key to be any other symmetric or asymmetric key in the
             keystore.";
          uses encrypted-by-choice-grouping;
        }
      }
    }

  }
```

   <CODE ENDS>

## 3.  Support for Built-in Keys

In some implementations, a server may support built-in keys. Built-in keys MAY be set during the manufacturing process or be dynamically generated the first time the server is booted or a particular service (e.g., SSH) is enabled.

The primary characteristic of the built-in keys is that they are provided by the system, as opposed to configuration. As such, they are present in <operational>. The example below illustrates what the keystore in <operational> might look like for a server in its factory default state.

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  or:origin="or:intended">
  <asymmetric-keys>
    <asymmetric-key or:origin="or:system">
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

In order for the built-in keys (and/or their associated built-in certificates) to be referenced by configuration, the referenced keys MUST first be copied into <running>. The keys SHOULD be copied into <running> using the same value for the list's "key" substatement, so that the server can bind the references to the built-in entries.

In addition to copying keys into the Keystore in <running>, cleartext and encrypted keys may be copied into other parts of configuration, but they will lose their connection to having been a built-in value. Note that hidden keys cannot be copied into other parts of the configuration because doing would lose the key's connection to the built-in key, where the key's secret value is stored. Built-in "encrypted" keys MAY be copied into other parts of the configuration so long as the reference to the other built-in key that encrypted them is maintained.

Only the referenced keys need to be copied; that is, the keys in <running> MAY be a subset, including the whole of the set, of the built-in keys defined in <operational>.

No new built-in keys may be added nor existing built-in changed, with exception for associating additional certificates to an existing built-in key.

A server MUST reject attempts to modify any aspect of built-in keys, with exception for associating additional certificates to a built-in key. That these keys are "configured" in <running> is an illusion, as they are strictly a read-only subset of that which must already exist in <operational>.

The following example illustrates how a single built-in key definition from the previous example has been propagated to <running>:

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <asymmetric-keys>
    <asymmetric-key>
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
        <certificate>
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

After the above configuration is applied, <operational> should appear as follows:

```
<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
  xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin"
  or:origin="or:intended">
  <asymmetric-keys>
    <asymmetric-key or:origin="or:system">
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>
        ct:subject-public-key-info-format
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
        <certificate or:origin="or:intended">
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>base64encodedvalue==</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

## 4. Encrypting Keys in Configuration

This section describes an approach that enables both the symmetric and asymmetric keys on a server to be encrypted, such that traditional backup/restore procedures can be used without concern for the keys being compromised when in transit.

### 4.1. Key Encryption Key

The ability to encrypt configured keys is predicated on the existence of a "key encryption key" (KEK). There may be any number of KEKs in a system. A KEK, by its namesake, is a key that is used to encrypt other keys. A KEK MAY be either a symmetric key or an asymmetric key.

If a KEK is a symmetric key, then the server MUST provide an API for administrators to encrypt other keys without needing to know the symmetric key's value. If the KEK is an asymmetric key, then the server MAY provide an API enabling the encryption of other keys or, alternatively, let the administrators do so themselves using the asymmetric key's public half.

A server MUST possess (or be able to possess, in case the KEK has been encrypted by another KEK) a KEK's cleartext value so that it can decrypt the other keys in the configurion at runtime.

## 4.2. Configuring Encrypted Keys

Each time a new key is configured, it SHOULD be encrypted by a KEK.

In "ietf-crypto-types" [I-D.ietf-netconf-crypto-types], the format for an encrypted symmetric key is described by the "encrypted-one-symmetric-key-format" identity, while the format for an encrypted asymmetric key is described by the "encrypted-one-asymmetric-key-format" identity

Implementations SHOULD provide an API that simultaneously generates and encrypts a key (symmetric or asymmetric) using a KEK. Thusly newly generated key cleartext values are never known to the administrators generating the keys.

In case the server implementation does not provide such an API, then the generating and encrypting steps MAY be performed outside the server, e.g., by an administrator with special access control rights (e.g., an organization's crypto officer).

In either case, the encrypted key can be configured into the Keystore using either the "encrypted-key" (for symmetric keys) or the "encrypted-private-key" (for asymmetric keys) nodes. These two nodes contain both the encrypted value as well as a reference to the KEK that encrypted the key.

## 4.3. Migrating Configuration to Another Server

When a KEK is used to encrypt other keys, migrating the configuration to another server is only possible if the second server has the same KEK. How the second server comes to have the same KEK is discussed in this section.

In some deployments, mechanisms outside the scope of this document may be used to migrate a KEK from one server to another. That said, beware that the ability to do so typically entails having access to the first server but, in many RMA scenarios, the first server may no longer be operational.

In other deployments, an organization's crypto officer, possessing a KEK's cleartext value, configures the same KEK on the second server, presumably as a hidden key or a key protected by access-control (e.g., NACM's "default-deny-all), so that the cleartext value is not disclosed to regular administrators. However, this approach creates high-coupling to and dependency on the crypto officers that doesn't scale in production environments.

In order to decouple the crypto officers from the regular administrators, a special KEK, called the "master key" (MK), may be used.

A MK is commonly a globally-unique built-in (see Section 3) asymmetric key. The private key, due to its long lifetime, is hidden (i.e., "hidden-private-key" in Section 2.1.4.5. of [I-D.ietf-netconf-crypto-types]). The public key is often contained in an identity certificate (e.g., IDevID). How to configure a MK during the manufacturing process is outside the scope of this document.

It is highly RECOMMENDED that MKs are built-in and hidden but, if this is not possible, MKs highly restricted access mechanisms SHOULD be used to limit access to the MK's secret data to only highly authorized clients (e.g., an organization's crypto officer). In this case, it is RECOMMENDED that the MK is not built-in and hence is, effectively, just like a KEK.

Assuming the server has a MK, the MK can be used to encrypt a "shared KEK", which is then used to encrypt the keys configured by regular administrators.

With this extra level of indirection, it is possible for a crypto officer to encrypt the same KEK for a multiplicity of servers offline using the public key contained in their identity certificates. The crypto officer can then safely handoff the encrypted KEKs to the regular administrators responsible for server installations, including migrations.

In order to migrate the configuration from a first server, an administrator would need to make just a single modification to the configuration before loading it onto a second server, which is to replace the encrypted KEK Keystore entry from the first server with the encrypted KEK for the second server. Upon doing this, the configuration (containing many encrypted keys) can be loaded into the second server while enabling the second server to decrypt all the encrypted keys in the configuration.

The following diagram illustrates this idea:

```
 +-------------+                                 +-------------+
 | shared KEK  |                                 | shared KEK  |
 |(unencrypted)|-------------------------------> | (encrypted) |
 +-------------+     encrypts offline using      +-------------+
        ^            each server's MK                 |
        |                                             |
        |                                             |
        |  possesses    \o                            |
        +--------------  |\                           |
                        / \         shares with       |
                      crypto    +---------------------+
                      officer   |
                                |
                                |
+----------------------+        |         +----------------------+
|       server-1       |        |         |       server-2       |
|    configuration     |        |         |    configuration     |
|                      |        |         |                      |
|                      |        |         |                      |
|  +----------------+  |        |         |  +----------------+  |
|  |      MK-1      |  |        |         |  |      MK-2      |  |
|  |    (hidden)    |  |        |         |  |    (hidden)    |  |
|  +----------------+  |        |         |  +----------------+  |
|      ^               |        |         |      ^               |
|      |               |        |         |      |               |
|      |               |        |         |      |               |
|      |  encrypted    |        |         |      |  encrypted    |
|      |  by           |        |         |      |  by           |
|      |               |        |         |      |               |
|      |               |        |         |      |               |
|  +----------------+  |        |         |  +----------------+  |
|  |  shared KEK    |  |        |         |  |  shared KEK    |  |
|  |  (encrypted)   |  |        v         |  |  (encrypted)   |  |
|  +----------------+  |                  |  +----------------+  |
|      ^               |     regular      |      ^               |
|      |               |      admin       |      |               |
|      |               |                  |      |               |
|      |  encrypted    |       \o         |      |  encrypted    |
|      |  by           |        |\        |      |  by           |
|      |               |       / \        |      |               |
|      |               |                  |      |               |
|  +----------------+  |----------------->|  +----------------+  |
|  | all other keys |  |     migrate      |  | all other keys |  |
|  |  (encrypted)   |  |  configuration   |  |  (encrypted)   |  |
|  +----------------+  |                  |  +----------------+  |
|                      |                  |                      |
+----------------------+                  +----------------------+
```

## 5. Security Considerations

### 5.1. Data at Rest

The YANG module defined in this document defines a mechanism called a "keystore" that, by its name, suggests that it will protect its contents from unauthorized disclosure and modification.

Security controls for the API (i.e., data in motion) are discussed in Section 5.2, but controls for the data at rest cannot be specified by the YANG module.

In order to satisfy the expectations of a "keystore", it is RECOMMENDED that implementations ensure that the keystore contents are encrypted when persisted to non-volatile memory.

### 5.2. The "ietf-keystore" YANG Module

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.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

Please be aware that this module uses the "cleartext-key" and "cleartext-private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing uncontrolled read-access to the cleartext key values.

All of the writable data nodes defined by this module, both in the "grouping" statements as well as the protocol-accessible "keystore" instance, may be considered sensitive or vulnerable in some network environments.. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

## 6. IANA Considerations

### 6.1. 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.
```

### 6.2. 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
namespace:    urn:ietf:params:xml:ns:yang:ietf-keystore
prefix:       ks
reference:    RFC CCCC
```

## 7. References

### 7.1. Normative References

**[I-D.ietf-netconf-crypto-types]**
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-17, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-crypto-types-17>.

**[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>.

## 7.2. Informative References

**[I-D.ietf-netconf-http-client-server]**
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-04, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-http-client-server-04>.

**[I-D.ietf-netconf-keystore]** Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-19, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-keystore-19>.

**[I-D.ietf-netconf-netconf-client-server]**
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server-20>.

**[I-D.ietf-netconf-restconf-client-server]**
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-20, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server-20>.

**[I-D.ietf-netconf-ssh-client-server]**
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server-21>.

**[I-D.ietf-netconf-tcp-client-server]**
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-07, 8 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server-07>.

**[I-D.ietf-netconf-tls-client-server]**
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-21, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-21>.

**[I-D.ietf-netconf-trust-anchors]**
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-12, 10 July 2020, <https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors-12>.

**[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

This section is to be removed before publishing as an RFC.

### 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.

## 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.

## A.13. 12 to 13

* Updated examples to incorporate new "key-format" identities.

* Made the two "generate-*-key" RPCs be "action" statements instead.

## A.14. 13 to 14

* Updated YANG module and examples to incorporate the new iana-*-algorithm modules in the crypto-types draft..

## A.15. 14 to 15

* Added new "Support for Built-in Keys" section.

* Added 'must' expressions asserting that the 'key-format' leaf whenever an encrypted key is specified.

* Added local-or-keystore-symmetric-key-grouping for PSK support.

## A.16. 15 to 16

* Moved the generate key actions to ietf-crypt-types as RPCs, which are augmented by ietf-keystore to support encrypted keys. Examples updated accordingly.

* Added a SSH certificate-based key (RFC 6187) and a raw private key to the example instance document (partly so they could be referenced by examples in the SSH and TLS client/server drafts.

## A.17. 16 to 17

* Removed augments to the "generate-symmetric-key" and "generate-asymmetric-key" groupings.

* Removed "generate-symmetric-key" and "generate-asymmetric-key" examples.

* Removed the "algorithm" nodes from remaining examples.

* Updated the "Support for Built-in Keys" section.

* Added new section "Encrypting Keys in Configuration".

* Added a "Note to Reviewers" note to first page.

## A.18. 17 to 18

* Removed dangling/unnecessary ref to RFC 8342.

* r/MUST/SHOULD/ wrt strength of keys being configured over transports.

* Added an example for the "certificate-expiration" notification.

* Clarified that OS MAY have a multiplicity of underlying keystores and/or HSMs.

* Clarified expected behavior for "built-in" keys in &lt;operational&gt;

* Clarified the "Migrating Configuration to Another Server" section.

*Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

*Updated the Security Considerations section.

### A.19. 18 to 19

*Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

### A.20. 19 to 20

*Addressed SecDir comments from Magnus Nystroem and Sandra Murphy.

## 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, Magnus Nystroem, Martin Bjorklund, Mehmet Ersue, Phil Shafer, Radek Krejci, Ramkumar Dhanapal, Reshad Rahman, Sandra Murphy, Sean Turner, and Tom Petch.

## Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net