Stream: Internet Engineering Task Force (IETF)
RFC: 9642
Category: Standards Track
Published: October 2024
ISSN: 2070-1721
Author: K. Watsen
Watsen Networks

RFC 9642
A YANG Data Model for a Keystore

Abstract

This document presents a YANG 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.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9642.

Copyright Notice

Table of Contents

1. Introduction

This document presents 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 [RFC9640]). 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 configured either inline (within the defining data model) or as a reference to a key in the central keystore.

Special consideration has been given for servers that have cryptographic hardware, such as a trusted platform module (TPM). These servers are unique in that the cryptographic hardware hides the secret key values. Additionally, such hardware is commonly initialized when manufactured to protect a "built-in" asymmetric key for which its public half is conveyed in an identity certificate (e.g., an Initial Device Identifier (IDevID) [Std-802.1AR-2018] certificate). See how built-in keys are supported in Section 3.

This document is intended to reflect existing practices that many server implementations support at the time of writing. To simplify implementation, advanced key formats may be selectively implemented.

Implementations may utilize operating-system level keystore utilities (e.g., "Keychain Access" on MacOS) and/or cryptographic hardware (e.g., TPMs).

1.1. Relation to Other RFCs

This document presents a YANG module [RFC7950] that is part of a collection of RFCs that work together to ultimately support the configuration of both the clients and servers of the Network Configuration Protocol (NETCONF) [RFC6241] and RESTCONF [RFC8040].

The dependency relationship between the primary YANG groupings defined in the various RFCs is presented in the diagram below. In some cases, a document may define secondary groupings that introduce dependencies not illustrated in the diagram. The labels in the diagram are shorthand names for the defining RFCs. The citation references for the shorthand names are provided below the diagram.

Please note that the arrows in the diagram point from referencer to referenced. For example, the "crypto-types" RFC does not have any dependencies, whilst the "keystore" RFC depends on the "crypto-types" RFC.

```
                               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	[RFC9640]
truststore	[RFC9641]
keystore	RFC 9642
tcp-client-server	[RFC9643]
ssh-client-server	[RFC9644]
tls-client-server	[RFC9645]
http-client-server	[HTTP-CLIENT-SERVER]
netconf-client-server	[NETCONF-CLIENT-SERVER]
restconf-client-server	[RESTCONF-CLIENT-SERVER]

Table 1: Labels in Diagram 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 document as a mechanism that intends to safeguard secrets.

The nomenclatures "<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].

The term "key" may be used to mean one of three things in this document: 1) the YANG-defined "asymmetric-key" or "symmetric-key" node defined in this document, 2) the raw key data possessed by the aforementioned key nodes, or 3) the "key" of a YANG "list" statement. This document qualifies types '2' and '3' using "raw key value" and "YANG list key" where needed. In all other cases, an unqualified "key" refers to a YANG-defined "asymmetric-key" or "symmetric-key" node.

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).

1.5. Conventions

Various examples in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (per Section 9.8 of [RFC7950]). This placeholder value is used because real base64-encoded structures are often many lines long and hence distracting to the example being presented.

Various examples in this document use the XML [W3C.REC-xml-20081126] encoding. Other encodings, such as JSON [RFC8259], could alternatively be used.

Various examples in this document contain long lines that may be folded, as described in [RFC8792].

This document uses the adjective "central" to the word "keystore" to refer to the top-level instance of the "keystore-grouping", when the "central-keystore-supported" feature is enabled. Please be aware that consuming YANG modules **MAY** instantiate the "keystore-grouping" in other locations. All such other instances are not the "central" instance.

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 illustrating the module's use are provided in 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:
  +-- central-keystore-supported
  +-- inline-definitions-supported
  +-- asymmetric-keys
  +-- symmetric-keys
```

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
    +-- central-symmetric-key-ref
    +-- central-asymmetric-key-ref
```

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

Comments:

- All 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 central keystore when this module is implemented.
- These typedefs are provided as an aid to consuming modules that import the "ietf-keystore" module.

2.1.3. Groupings

The "ietf-keystore" module defines the following "grouping" statements:

- encrypted-by-grouping
- central-asymmetric-key-certificate-ref-grouping
- inline-or-keystore-symmetric-key-grouping
- inline-or-keystore-asymmetric-key-grouping
- inline-or-keystore-asymmetric-key-with-certs-grouping
- inline-or-keystore-end-entity-cert-with-key-grouping
- keystore-grouping

Each of these groupings are presented in the following subsections.

2.1.3.1. The "encrypted-by-grouping" Grouping

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

```
grouping encrypted-by-grouping:
  +-- (encrypted-by)
     +--:(central-symmetric-key-ref)
     |        {central-keystore-supported,symmetric-keys}?
     |  +-- symmetric-key-ref?    ks:central-symmetric-key-ref
     +--:(central-asymmetric-key-ref)
              {central-keystore-supported,asymmetric-keys}?
        +-- asymmetric-key-ref?   ks:central-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.
- This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations **MUST** augment in alternate "encrypted-by" references to the alternate locations.

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

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

```
grouping central-asymmetric-key-certificate-ref-grouping:
  +-- asymmetric-key?   ks:central-asymmetric-key-ref
  |       {central-keystore-supported,asymmetric-keys}?
  +-- 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.
- This grouping is usable only when the keystore module is implemented. Servers defining custom keystore locations can define an alternate grouping for references to the alternate locations.

2.1.3.3. The "inline-or-keystore-symmetric-key-grouping" Grouping

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

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

Comments:

- The "inline-or-keystore-symmetric-key-grouping" grouping is provided solely as convenience to consuming modules that wish to offer an option for a symmetric key that is defined either inline 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 "inline-definition" option, the definition uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [RFC9640].
- For the "central-keystore" option, the "central-keystore-reference" is an instance of the "symmetric-key-ref" discussed in Section 2.1.2.

2.1.3.4. The "inline-or-keystore-asymmetric-key-grouping" Grouping

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

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

Comments:

- The "inline-or-keystore-asymmetric-key-grouping" grouping is provided solely as convenience to consuming modules that wish to offer an option for an asymmetric key that is defined either inline 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 "inline-definition" option, the definition uses the "asymmetric-key-pair-grouping" grouping discussed in Section 2.1.4.6 of [RFC9640].
- For the "central-keystore" option, the "central-keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

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

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

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

Comments:

- The "inline-or-keystore-asymmetric-key-with-certs-grouping" grouping is provided solely as convenience to consuming modules that wish to offer an option for an asymmetric key that is defined either inline 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 "inline-definition" option, the definition uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.12 of [RFC9640].
- For the "central-keystore" option, the "central-keystore-reference" is an instance of the "asymmetric-key-ref" typedef discussed in Section 2.1.2.

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

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

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

Comments:

- The "inline-or-keystore-end-entity-cert-with-key-grouping" grouping is provided solely as convenience to consuming modules that wish to offer an option for a symmetric key that is defined either inline 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 "inline-definition" option, the definition uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.12 of [RFC9640].
- For the "central-keystore" option, the "central-keystore-reference" uses the "central-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-keys}?
  |  +-- asymmetric-key* [name]
  |     +-- name                                          string
  |     +---u ct:asymmetric-key-pair-with-certs-grouping
  +-- symmetric-keys {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 structure for the symmetric and asymmetric keys is essentially the same: a "list" inside a "container".
- For asymmetric keys, each "asymmetric-key" uses the "asymmetric-key-pair-with-certs-grouping" grouping discussed in Section 2.1.4.12 of [RFC9640].
- For symmetric keys, each "symmetric-key" uses the "symmetric-key-grouping" grouping discussed in Section 2.1.4.3 of [RFC9640].

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 {central-keystore-supported}?
     +---u keystore-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.

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

module: ietf-keystore
  +--rw keystore {central-keystore-supported}?
     +--rw asymmetric-keys {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) {cleartext-private-keys}?
     |     |  |  +--rw cleartext-private-key?   binary
     |     |  +--:(hidden-private-key) {hidden-private-keys}?
     |     |  |  +--rw hidden-private-key?      empty
     |     |  +--:(encrypted-private-key) {encrypted-private-keys}?
     |     |     +--rw encrypted-private-key
     |     |        +--rw encrypted-by
     |     |        |  +--rw (encrypted-by)
     |     |        |     +--:(central-symmetric-key-ref)
     |     |        |     |        {central-keystore-supported,symme\
tric-keys}?
     |     |        |     |  +--rw symmetric-key-ref?
     |     |        |     |          ks:central-symmetric-key-ref
     |     |        |     +--:(central-asymmetric-key-ref)
     |     |        |              {central-keystore-supported,asymm\
etric-keys}?
     |     |        |        +--rw asymmetric-key-ref?
     |     |        |                ks:central-asymmetric-key-ref
     |     |        +--rw encrypted-value-format    identityref
     |     |        +--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-csr {csr-generation}?
     |        +---w input
     |        |  +---w csr-format    identityref
     |        |  +---w csr-info      csr-info
     |        +--ro output
     |           +--ro (csr-type)
     |              +--:(p10-csr)
     |                 +--ro p10-csr?   p10-csr
     +--rw symmetric-keys {symmetric-keys}?
        +--rw symmetric-key* [name]
           +--rw name                             string
           +--rw key-format?                      identityref
           +--rw (key-type)
              +--:(cleartext-symmetric-key)
              |  +--rw cleartext-symmetric-key?   binary
              |          {cleartext-symmetric-keys}?
              +--:(hidden-symmetric-key) {hidden-symmetric-keys}?
              |  +--rw hidden-symmetric-key?      empty
              +--:(encrypted-symmetric-key)
                       {encrypted-symmetric-keys}?
                 +--rw encrypted-symmetric-key
                    +--rw encrypted-by
                    |  +--rw (encrypted-by)
                    |     +--:(central-symmetric-key-ref)
                    |     |        {central-keystore-supported,symme\
tric-keys}?
                    |     |  +--rw symmetric-key-ref?
                    |     |          ks:central-symmetric-key-ref
                    |     +--:(central-asymmetric-key-ref)
                    |              {central-keystore-supported,asymm\
etric-keys}?
                    |        +--rw asymmetric-key-ref?
                    |                ks:central-asymmetric-key-ref
                    +--rw encrypted-value-format    identityref
                    +--rw 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 protocol-accessible nodes for the "ietf-keystore" module are instances of the "keystore-grouping" grouping discussed in Section 2.1.3.7.
- The top-level node "keystore" is additionally constrained by the feature "central-keystore-supported".
- The "keystore-grouping" grouping is discussed in Section 2.1.3.7.

- The reason for why "keystore-grouping" exists separate from the protocol-accessible nodes definition is 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-symmetric-key>BASE64VALUE=</cleartext-symmetric-\
key>
      </symmetric-key>
      <symmetric-key>
         <name>hidden-symmetric-key</name>
         <hidden-symmetric-key/>
      </symmetric-key>
      <symmetric-key>
         <name>encrypted-symmetric-key</name>
         <key-format>ct:one-symmetric-key-format</key-format>
         <encrypted-symmetric-key>
           <encrypted-by>
             <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-k\
ey-ref>
           </encrypted-by>
           <encrypted-value-format>ct:cms-enveloped-data-format</enc\
rypted-value-format>
           <encrypted-value>BASE64VALUE=</encrypted-value>
         </encrypted-symmetric-key>
      </symmetric-key>
   </symmetric-keys>

   <asymmetric-keys>
      <asymmetric-key>
         <name>ssh-rsa-key</name>
         <private-key-format>ct:rsa-private-key-format</private-key-\
format>
         <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      </asymmetric-key>
      <asymmetric-key>
```

```
         <name>ssh-rsa-key-with-cert</name>
         <private-key-format>ct:rsa-private-key-format</private-key-\
format>
         <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         <certificates>
            <certificate>
               <name>ex-rsa-cert2</name>
               <cert-data>BASE64VALUE=</cert-data>
            </certificate>
         </certificates>
      </asymmetric-key>
      <asymmetric-key>
         <name>raw-private-key</name>
         <private-key-format>ct:rsa-private-key-format</private-key-\
format>
         <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      </asymmetric-key>
      <asymmetric-key>
         <name>rsa-asymmetric-key</name>
         <private-key-format>ct:rsa-private-key-format</private-key-\
format>
         <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         <certificates>
            <certificate>
               <name>ex-rsa-cert</name>
               <cert-data>BASE64VALUE=</cert-data>
            </certificate>
         </certificates>
      </asymmetric-key>
      <asymmetric-key>
         <name>ec-asymmetric-key</name>
         <private-key-format>ct:ec-private-key-format</private-key-f\
ormat>
         <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
         <certificates>
            <certificate>
               <name>ex-ec-cert</name>
               <cert-data>BASE64VALUE=</cert-data>
            </certificate>
         </certificates>
      </asymmetric-key>
      <asymmetric-key>
         <name>hidden-asymmetric-key</name>
         <public-key-format>ct:subject-public-key-info-format</publi\
c-key-format>
         <public-key>BASE64VALUE=</public-key>
         <hidden-private-key/>
         <certificates>
            <certificate>
               <name>builtin-idevid-cert</name>
               <cert-data>BASE64VALUE=</cert-data>
            </certificate>
            <certificate>
               <name>my-ldevid-cert</name>
               <cert-data>BASE64VALUE=</cert-data>
            </certificate>
         </certificates>
      </asymmetric-key>
      <asymmetric-key>
         <name>encrypted-asymmetric-key</name>
         <private-key-format>ct:one-asymmetric-key-format</private-k\
ey-format>
         <encrypted-private-key>
           <encrypted-by>
             <symmetric-key-ref>encrypted-symmetric-key</symmetric-k\
ey-ref>
           </encrypted-by>
           <encrypted-value-format>ct:cms-encrypted-data-format</enc\
rypted-value-format>
           <encrypted-value>BASE64VALUE=</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 an asymmetric 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 "Inline or Keystore" Groupings

This section illustrates the various "inline-or-keystore" groupings defined in the "ietf-keystore" module, specifically the "inline-or-keystore-symmetric-key-grouping" (Section 2.1.3.3), "inline-or-keystore-asymmetric-key-grouping" (Section 2.1.3.4), "inline-or-keystore-asymmetric-key-with-certs-grouping" (Section 2.1.3.5), and "inline-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" that has the namespace "https://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 "inline-or-keystore" groupings in use, followed by the YANG module itself.

2.2.3.1. Tree Diagrams for the "ex-keystore-usage" Module

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

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

module: ex-keystore-usage
  +--rw keystore-usage
     +--rw symmetric-key* [name]
     |  +--rw name                                            string
     |  +---u ks:inline-or-keystore-symmetric-key-grouping
     +--rw asymmetric-key* [name]
     |  +--rw name                                             string
     |  +---u ks:inline-or-keystore-asymmetric-key-grouping
     +--rw asymmetric-key-with-certs* [name]
     |  +--rw name
     |  |       string
     |  +---u ks:inline-or-keystore-asymmetric-key-with-certs-groupi\
ng
     +--rw end-entity-cert-with-key* [name]
        +--rw name
        |       string
        +---u ks:inline-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 usage's full structure to be seen:

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

module: ex-keystore-usage
  +--rw keystore-usage
     +--rw symmetric-key* [name]
     |  +--rw name                                string
     |  +--rw (inline-or-keystore)
     |     +--:(inline) {inline-definitions-supported}?
     |     |  +--rw inline-definition
     |     |     +--rw key-format?                      identityref
     |     |     +--rw (key-type)
     |     |        +--:(cleartext-symmetric-key)
     |     |        |  +--rw cleartext-symmetric-key?   binary
     |     |        |          {cleartext-symmetric-keys}?
     |     |        +--:(hidden-symmetric-key)
     |     |        |        {hidden-symmetric-keys}?
     |     |        |  +--rw hidden-symmetric-key?      empty
     |     |        +--:(encrypted-symmetric-key)
```

```
      |     |                 {encrypted-symmetric-keys}?
      |     |           +--rw encrypted-symmetric-key
      |     |              +--rw encrypted-by
      |     |              +--rw encrypted-value-format    identityref
      |     |              +--rw encrypted-value           binary
      |     +--:(central-keystore)
      |              {central-keystore-supported,symmetric-keys}?
      |        +--rw central-keystore-reference?
      |                ks:central-symmetric-key-ref
      +--rw asymmetric-key* [name]
      |  +--rw name                                string
      |  +--rw (inline-or-keystore)
      |     +--:(inline) {inline-definitions-supported}?
      |     |  +--rw inline-definition
      |     |     +--rw public-key-format?             identityref
      |     |     +--rw public-key?                    binary
      |     |     +--rw private-key-format?            identityref
      |     |     +--rw (private-key-type)
      |     |        +--:(cleartext-private-key)
      |     |        |        {cleartext-private-keys}?
      |     |        |  +--rw cleartext-private-key?   binary
      |     |        +--:(hidden-private-key) {hidden-private-keys}?
      |     |        |  +--rw hidden-private-key?      empty
      |     |        +--:(encrypted-private-key)
      |     |                 {encrypted-private-keys}?
      |     |           +--rw encrypted-private-key
      |     |              +--rw encrypted-by
      |     |              +--rw encrypted-value-format    identityref
      |     |              +--rw encrypted-value           binary
      |     +--:(central-keystore)
      |              {central-keystore-supported,asymmetric-keys}?
      |        +--rw central-keystore-reference?
      |                ks:central-asymmetric-key-ref
      +--rw asymmetric-key-with-certs* [name]
      |  +--rw name                                string
      |  +--rw (inline-or-keystore)
      |     +--:(inline) {inline-definitions-supported}?
      |     |  +--rw inline-definition
      |     |     +--rw public-key-format?             identityref
      |     |     +--rw public-key?                    binary
      |     |     +--rw private-key-format?            identityref
      |     |     +--rw (private-key-type)
      |     |     |  +--:(cleartext-private-key)
      |     |     |  |        {cleartext-private-keys}?
      |     |     |  |  +--rw cleartext-private-key?   binary
      |     |     |  +--:(hidden-private-key) {hidden-private-keys}?
      |     |     |  |  +--rw hidden-private-key?      empty
      |     |     |  +--:(encrypted-private-key)
      |     |     |           {encrypted-private-keys}?
      |     |     |     +--rw encrypted-private-key
      |     |     |        +--rw encrypted-by
      |     |     |        +--rw encrypted-value-format    identityref
      |     |     |        +--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-csr {csr-generation}?
     |     |        +---w input
     |     |        |  +---w csr-format    identityref
     |     |        |  +---w csr-info      csr-info
     |     |        +--ro output
     |     |           +--ro (csr-type)
     |     |              +--:(p10-csr)
     |     |                 +--ro p10-csr?   p10-csr
     |     +--:(central-keystore)
     |              {central-keystore-supported,asymmetric-keys}?
     |        +--rw central-keystore-reference?
     |                ks:central-asymmetric-key-ref
     +--rw end-entity-cert-with-key* [name]
        +--rw name                                string
        +--rw (inline-or-keystore)
           +--:(inline) {inline-definitions-supported}?
           |  +--rw inline-definition
           |     +--rw public-key-format?             identityref
           |     +--rw public-key?                    binary
           |     +--rw private-key-format?            identityref
           |     +--rw (private-key-type)
           |     |  +--:(cleartext-private-key)
           |     |  |        {cleartext-private-keys}?
           |     |  |  +--rw cleartext-private-key?   binary
           |     |  +--:(hidden-private-key) {hidden-private-keys}?
           |     |  |  +--rw hidden-private-key?      empty
           |     |  +--:(encrypted-private-key)
           |     |           {encrypted-private-keys}?
           |     |     +--rw encrypted-private-key
           |     |        +--rw encrypted-by
           |     |        +--rw encrypted-value-format    identityref
           |     |        +--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-csr {csr-generation}?
           |        +---w input
           |        |  +---w csr-format    identityref
           |        |  +---w csr-info      csr-info
           |        +--ro output
           |           +--ro (csr-type)
           |              +--:(p10-csr)
           |                 +--ro p10-csr?   p10-csr
           +--:(central-keystore)
                    {central-keystore-supported,asymmetric-keys}?
              +--rw central-keystore-reference
                 +--rw asymmetric-key?
                 |       ks:central-asymmetric-key-ref
                 |       {central-keystore-supported,asymmetric-keys\
}?
                 +--rw certificate?      leafref
```

2.2.3.2. Example Usage for the "ex-keystore-usage" Module

The following example provides two equivalent instances of each grouping, the first being a reference to a keystore and the second being inlined. 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 inlined 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="https://example.com/ns/example-keystore-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

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

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

  <symmetric-key>
    <name>example 1b</name>
    <inline-definition>
      <key-format>ct:octet-string-key-format</key-format>
      <cleartext-symmetric-key>BASE64VALUE=</cleartext-symmetric-key>
    </inline-definition>
  </symmetric-key>


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

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

  <asymmetric-key>
    <name>example 2b</name>
    <inline-definition>
      <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
    </inline-definition>
  </asymmetric-key>


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

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

  <asymmetric-key-with-certs>
    <name>example 3b</name>
    <inline-definition>
      <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      <certificates>
        <certificate>
          <name>a locally defined cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </inline-definition>
  </asymmetric-key-with-certs>


  <!-- The following two equivalent examples illustrate the    -->
  <!-- "inline-or-keystore-end-entity-cert-with-key-grouping"  -->
  <!-- grouping:                                               -->
  <end-entity-cert-with-key>
    <name>example 4a</name>
    <central-keystore-reference>
      <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
      <certificate>ex-rsa-cert</certificate>
    </central-keystore-reference>
  </end-entity-cert-with-key>

  <end-entity-cert-with-key>
    <name>example 4b</name>
    <inline-definition>
      <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      <cert-data>BASE64VALUE=</cert-data>
    </inline-definition>
  </end-entity-cert-with-key>

</keystore-usage>
```

2.2.3.3. The "ex-keystore-usage" YANG Module

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

```
module ex-keystore-usage {
  yang-version 1.1;
  namespace "https://example.com/ns/example-keystore-usage";
  prefix ex-keystore-usage;

  import ietf-keystore {
    prefix ks;
    reference
      "RFC 9642: A YANG Data Model for a Keystore";
  }

  organization
    "Example Corporation";

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

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

  revision 2024-03-16 {
    description
      "Initial version";
    reference
      "RFC 9642: 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:inline-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:inline-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:inline-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:inline-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 [RFC9640].

```
<CODE BEGINS> file "ietf-keystore@2024-03-16.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 9640: YANG Data Types and Groupings for Cryptography";
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
     WG List:  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.

     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.

     Copyright (c) 2024 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 Revised
     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 9642
     (https://www.rfc-editor.org/info/rfc9642); see the RFC
     itself for full legal notices.";

  revision 2024-03-16 {
    description
      "Initial version";
    reference
      "RFC 9642: A YANG Data Model for a Keystore";
  }

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

  feature central-keystore-supported {
    description
      "The 'central-keystore-supported' feature indicates that
       the server supports the central keystore (i.e., fully
       implements the 'ietf-keystore' module).";
  }

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

  feature asymmetric-keys {
    description
      "The 'asymmetric-keys' feature indicates that the server
```

```
       implements the /keystore/asymmetric-keys subtree.";

  }

  feature symmetric-keys {
    description
      "The 'symmetric-keys' feature indicates that the server
       implements the /keystore/symmetric-keys subtree.";
  }

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

  typedef central-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 central keystore.";
  }

  typedef central-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 central keystore.";
  }

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

  grouping encrypted-by-grouping {
    description
      "A grouping that defines a 'choice' statement that can be
       augmented into the 'encrypted-by' node, present in the
       'symmetric-key-grouping' and 'asymmetric-key-pair-grouping'
       groupings defined in RFC 9640, enabling references to keys
       in the central keystore.";
    choice encrypted-by {
      nacm:default-deny-write;
      mandatory true;
      description
        "A choice amongst other symmetric or asymmetric keys.";
      case central-symmetric-key-ref {
        if-feature "central-keystore-supported";
        if-feature "symmetric-keys";
        leaf symmetric-key-ref {
          type ks:central-symmetric-key-ref;
          description
            "Identifies the symmetric key used to encrypt the
             associated key.";
        }
      }
      case central-asymmetric-key-ref {
        if-feature "central-keystore-supported";
        if-feature "asymmetric-keys";
        leaf asymmetric-key-ref {
          type ks:central-asymmetric-key-ref;
          description
            "Identifies the asymmetric key whose public key
             encrypted the associated key.";
        }
      }
    }
  }

  // *-ref groupings

  grouping central-asymmetric-key-certificate-ref-grouping {
    description
      "A grouping for the reference to a certificate associated
       with an asymmetric key stored in the central keystore.";
    leaf asymmetric-key {
      nacm:default-deny-write;
      if-feature "central-keystore-supported";
      if-feature "asymmetric-keys";
      type ks:central-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.";
    }
  }

  // inline-or-keystore-* groupings

  grouping inline-or-keystore-symmetric-key-grouping {
    description
      "A grouping for the configuration of a symmetric key.  The
       symmetric key may be defined inline or as a reference to
       a symmetric key stored in the central keystore.

       Servers that wish to define alternate keystore locations
       SHOULD augment in custom 'case' statements enabling
       references to those alternate keystore locations.";
    choice inline-or-keystore {
      nacm:default-deny-write;
      mandatory true;
```

```
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case inline {
          if-feature "inline-definitions-supported";
          container inline-definition {
            description
              "A container to hold the local key definition.";
            uses ct:symmetric-key-grouping;
          }
        }
        case central-keystore {
          if-feature "central-keystore-supported";
          if-feature "symmetric-keys";
          leaf central-keystore-reference {
            type ks:central-symmetric-key-ref;
            description
              "A reference to a symmetric key that exists in
               the central keystore.";
          }
        }
      }
    }

    grouping inline-or-keystore-asymmetric-key-grouping {
      description
        "A grouping for the configuration of an asymmetric key.  The
         asymmetric key may be defined inline or as a reference to
         an asymmetric key stored in the central keystore.

         Servers that wish to define alternate keystore locations
         SHOULD augment in custom 'case' statements enabling
         references to those alternate keystore locations.";
      choice inline-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case inline {
          if-feature "inline-definitions-supported";
          container inline-definition {
            description
              "A container to hold the local key definition.";
            uses ct:asymmetric-key-pair-grouping;
          }
        }
        case central-keystore {
          if-feature "central-keystore-supported";
          if-feature "asymmetric-keys";
          leaf central-keystore-reference {
            type ks:central-asymmetric-key-ref;
            description
              "A reference to an asymmetric key that exists in
               the central keystore.  The intent is to reference
               just the asymmetric key without any regard for
               any certificates that may be associated with it.";
          }
        }
      }
    }

    grouping inline-or-keystore-asymmetric-key-with-certs-grouping {
      description
        "A grouping for the configuration of an asymmetric key and
         its associated certificates.  The asymmetric key and its
         associated certificates may be defined inline or as a
         reference to an asymmetric key (and its associated
         certificates) in the central keystore.

         Servers that wish to define alternate keystore locations
         SHOULD augment in custom 'case' statements enabling
         references to those alternate keystore locations.";
      choice inline-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case inline {
          if-feature "inline-definitions-supported";
          container inline-definition {
            description
              "A container to hold the local key definition.";
            uses ct:asymmetric-key-pair-with-certs-grouping;
          }
        }
        case central-keystore {
          if-feature "central-keystore-supported";
          if-feature "asymmetric-keys";
          leaf central-keystore-reference {
            type ks:central-asymmetric-key-ref;
            description
              "A reference to an asymmetric key (and all of its
               associated certificates) in the keystore, when
               this module is implemented.";
          }
        }
      }
    }

    grouping inline-or-keystore-end-entity-cert-with-key-grouping {
      description
        "A grouping for the configuration of an asymmetric key and
         its associated end-entity certificate.  The asymmetric key
         and its associated end-entity certificate may be defined
         inline or as a reference to an asymmetric key (and its
         associated end-entity certificate) in the central keystore.

         Servers that wish to define alternate keystore locations
         SHOULD augment in custom 'case' statements enabling
         references to those alternate keystore locations.";
      choice inline-or-keystore {
        nacm:default-deny-write;
        mandatory true;
        description
```

```
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
        case inline {
          if-feature "inline-definitions-supported";
          container inline-definition {
            description
              "A container to hold the local key definition.";
            uses ct:asymmetric-key-pair-with-cert-grouping;
          }
        }
        case central-keystore {
          if-feature "central-keystore-supported";
          if-feature "asymmetric-keys";
          container central-keystore-reference {
            uses central-asymmetric-key-certificate-ref-grouping;
            description
              "A reference to a specific certificate associated with
               an asymmetric key stored in the central keystore.";
          }
        }
      }
    }

    // the keystore grouping

    grouping keystore-grouping {
      description
        "A grouping definition enables use in other contexts.  If ever
         done, implementations MUST augment new 'case' statements
         into the various inline-or-keystore 'choice' statements to
         supply leafrefs to the model-specific location(s).";
      container asymmetric-keys {
        nacm:default-deny-write;
        if-feature "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 ct:asymmetric-key-pair-with-certs-grouping;
        }
      }
      container symmetric-keys {
        nacm:default-deny-write;
        if-feature "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 ct:symmetric-key-grouping;
      }
    }
  }

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

  container keystore {
    if-feature "central-keystore-supported";
    description
      "A central keystore containing 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-"
            + "symmetric-key/encrypted-symmetric-key/encrypted-by" {
        description
          "Augments in a choice statement enabling the encrypting
           key to be any other symmetric or asymmetric key in the
           central keystore.";
        uses encrypted-by-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
           central keystore.";
        uses encrypted-by-grouping;
      }
    }
  }
}

<CODE ENDS>
```

3. Support for Built-In Keys

In some implementations, a server may support keys built into the server. 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., Secure Shell (SSH)) is enabled.

Built-in keys are "hidden" keys expected to be set by a vendor-specific process. Any ability for operators to set and/or modify built-in keys is outside the scope of this document.

The primary characteristic of the built-in keys is that they are provided by the server, as opposed to being configured. As such, they are present in <operational> (Section 5.3 of [RFC8342]) and <system> [NETMOD-SYSTEM-CONFIG], if implemented.

The example below illustrates what the keystore in <operational> might look like for a server in its factory default state. Note that the built-in keys have the "or:origin" annotation value "or:system".

```
=============== 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"
  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-k\
ey-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

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

```
=============== 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">
  <asymmetric-keys>
    <asymmetric-key>
      <name>Manufacturer-Generated Hidden Key</name>
      <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate>
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</keystore>
```

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

```
=============== 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"
  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-k\
ey-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>Manufacturer-Generated IDevID Cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
        <certificate or:origin="or:intended">
          <name>Deployment-Specific LDevID Cert</name>
          <cert-data>BASE64VALUE=</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 backup/restore procedures can be used without concern for raw key data being compromised when in transit.

The approach presented in this section is not normative. This section answers how a configuration containing secrets that are encrypted by a built-in key (Section 3) can be backed up from one server and restored on a different server when each server has unique primary keys. The API defined by the "ietf-keystore" YANG module presented in this document is sufficient to support the workflow described in this section.

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 server. 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 **SHOULD** provide an API enabling the encryption of other keys or, alternatively, assume the administrators can do so themselves using the asymmetric key's public half.

A server **MUST** possess access to the KEK, or an API using the KEK, so that it can decrypt the other keys in the configuration at runtime.

4.2. Configuring Encrypted Keys

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

In the "ietf-crypto-types" module [RFC9640], the format for encrypted values is described by identity statements derived from the "symmetrically-encrypted-value-format" and "asymmetrically-encrypted-value-format" identity statements.

Implementations of servers implementing the "ietf-keystore" module **SHOULD** provide an API that simultaneously generates a key and encrypts the generated key using a KEK. Thus, the cleartext value of the newly generated key may never be known to the administrators generating the keys. Such an API is defined in the "ietf-ssh-common" and "ietf-tls-common" YANG modules defined in [RFC9644] and [RFC9645], respectively.

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 (such as an organization's crypto officer).

In either case, the encrypted key can be configured into the keystore using either the "encrypted-symmetric-key" (for symmetric keys) or the "encrypted-private-key" (for asymmetric keys) nodes. These two nodes contain both the encrypted raw key 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; however, in some 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, 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 does not scale in production environments.

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

A PK is commonly a globally unique built-in (see Section 3) asymmetric key. The private raw key value, due to its long lifetime, is hidden (i.e., "hidden-private-key"; see Section 2.1.4.5. of [RFC9640]). The raw public key value is often contained in an identity certificate (e.g., IDevID). How to configure a PK during the manufacturing process is outside the scope of this document.

Assuming the server has a PK, the PK 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 hand off the encrypted KEKs to 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 PK                |
        |                                            |
        |                                            |
        |  possesses    \o                           |
        +--------------  |\                          |
                        / \         shares with      |
                      crypto    +--------------------+
                      officer   |
                                |
                                |
+----------------------+        |         +----------------------+
|       server-1       |        |         |       server-2       |
|    configuration     |        |         |    configuration     |
|                      |        |         |                      |
|                      |        |         |                      |
|  +----------------+  |        |         |  +----------------+  |
|  |      PK-1      |  |        |         |  |      PK-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. Security of Data at Rest and in Motion

The YANG module defined in this document defines a mechanism called a "keystore" that intends to protect its contents from unauthorized disclosure and modification.

In order to satisfy the expectations of a keystore, it is **RECOMMENDED** that server implementations ensure that the keystore contents are encrypted when persisted to non-volatile memory and that the keystore contents that have been decrypted in volatile memory are zeroized when not in use.

The keystore contents may be encrypted by either encrypting the contents individually (e.g., using the "encrypted" value formats) or using persistence-layer-level encryption. If storing cleartext values (which is **NOT RECOMMENDED** per Section 3.5 of [RFC9640]), then persistence-layer-level encryption **SHOULD** be used to protect the data at rest.

If the keystore contents are not encrypted when persisted, then server implementations **MUST** ensure the persisted storage is inaccessible.

5.2. Unconstrained Private Key Usage

This module enables the configuration of private keys without constraints on their usage, e.g., what operations the key is allowed to be used for (such as signature, decryption, or both).

This module also does not constrain the usage of the associated public keys other than in the context of a configured certificate (e.g., an identity certificate), in which case the key usage is constrained by the certificate.

5.3. Security Considerations for the "ietf-keystore" YANG Module

This section is modeled after the template defined in Section 3.7.1 of [RFC8407].

The ietf-keystore YANG module defines a data model that is designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. These protocols have mandatory-to-implement secure transport layers (e.g., SSH [RFC4252], TLS [RFC8446], and QUIC [RFC9000]) and mandatory-to-implement mutual authentication.

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a preconfigured subset of all available protocol operations and content.

Please be aware that this YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

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:

The "cleartext-symmetric-key" node:
: This node, imported from the "symmetric-key-grouping" grouping defined in [RFC9640], is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" was applied to it in [RFC9640].

The "cleartext-private-key" node:
: This node, defined in the "asymmetric-key-pair-grouping" grouping in [RFC9640], is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" is applied to it in [RFC9640].

All the writable data nodes defined by this YANG 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 YANG module does not define any "rpc" or "action" statements, and thus the security considerations for such is not provided here.

Built-in key types **SHOULD** be hidden and/or encrypted (not cleartext). If this is not possible, access control mechanisms like NACM **SHOULD** be used to limit access to the key's secret data to only the most trusted authorized clients (e.g., belonging to an organization's crypto officer).

6. IANA Considerations

6.1. The IETF XML Registry

IANA has registered the following URI in the "ns" registry of the "IETF XML Registry" [RFC3688].

URI: urn:ietf:params:xml:ns:yang:ietf-keystore
Registrant Contact: The IESG
XML: N/A; the requested URI is an XML namespace.

6.2. The YANG Module Names Registry

IANA has registered the following YANG module in the "YANG Module Names" registry defined in [RFC6020].

Name: ietf-keystore
Maintained by IANA: N
Namespace: urn:ietf:params:xml:ns:yang:ietf-keystore
Prefix: ks
Reference: RFC 9642

7. References

7.1. Normative References

[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>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9640] Watsen, K., "YANG Data Types and Groupings for Cryptography", RFC 9640, DOI 10.17487/RFC9640, October 2024, <https://www.rfc-editor.org/info/rfc9640>.

7.2. Informative References

[HTTP-CLIENT-SERVER]

Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-23, 15 August 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-23>.

[NETCONF-CLIENT-SERVER] Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-37, 14 August 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-37>.

[NETMOD-SYSTEM-CONFIG] Ma, Q., Ed., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-09, 29 September 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-09>.

[RESTCONF-CLIENT-SERVER] Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-38, 14 August 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-38>.

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

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[RFC9641] Watsen, K., "A YANG Data Model for a Truststore", RFC 9641, DOI 10.17487/RFC9641, October 2024, <https://www.rfc-editor.org/info/rfc9641>.

[RFC9643] Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", RFC 9643, DOI 10.17487/RFC9643, October 2024, <https://www.rfc-editor.org/info/rfc9643>.

[RFC9644] Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", RFC 9644, DOI 10.17487/RFC9644, October 2024, <https://www.rfc-editor.org/info/rfc9644>.

[RFC9645] Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", RFC 9645, DOI 10.17487/RFC9645, October 2024, <https://www.rfc-editor.org/info/rfc9645>.

[Std-802.1AR-2018] IEEE, "IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity", IEEE Std 802.1AR-2018, DOI 10.1109/IEEESTD.2018.8423794, August 2018, <https://standards.ieee.org/standard/802_1AR-2018.html>.

[W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", W3C Recommendation REC-xml-20081126, November 2008, <https://www.w3.org/TR/xml/>.

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balázs Kovács, Benoit Claise, Bert Wijnen, David Lamparter, Eric Voit, Éric Vyncke, Francesca Palombini, Jürgen Schönwälder, Ladislav Lhotka, Liang Xia, Magnus Nyström, Mahesh Jethanandani, Martin Björklund, Mehmet Ersue, Murray Kucherawy, Paul Wouters, Phil Shafer, Qin Wu, Radek Krejci, Ramkumar Dhanapal, Reese Enghardt, Reshad Rahman, Rob Wilton, Roman Danyliw, Sandra Murphy, Sean Turner, Tom Petch, Warren Kumari, and Zaheduzzaman Sarker.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net