

NETCONF Working Group  
Watsen  
Internet-Draft  
Networks  
Intended status: Standards Track  
2017  
Expires: May 3, 2018

K.  
Juniper  
October 30,

**YANG Data Model for a "Keystore" Mechanism  
draft-ietf-netconf-keystore-04**

Abstract

This document defines a YANG module called a "keystore", containing pinned certificates and pinned SSH host-keys. The module also defines a grouping for configuring public key pairs and a grouping for configuring certificates. The module also defines a notification that a system can use when one of its configured certificates is about to expire.

Editorial Note (To be removed by RFC Editor)

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

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

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

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

- o "2017-10-30" --> the publication date of this draft

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

- o [Appendix A](#). Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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



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

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

- [1](#). Introduction . . . . . 3
- [1.1](#). Requirements Language . . . . . 3
- [2](#). Tree Diagram . . . . . 4
- [3](#). Example Usage . . . . . 5
- [4](#). YANG Module . . . . . 10
- [5](#). Security Considerations . . . . . 21
- [6](#). IANA Considerations . . . . . 22
  - [6.1](#). The IETF XML Registry . . . . . 22
  - [6.2](#). The YANG Module Names Registry . . . . . 22
- [7](#). Acknowledgements . . . . . 23
- [8](#). References . . . . . 23
  - [8.1](#). Normative References . . . . . 23
  - [8.2](#). Informative References . . . . . 24

[Appendix A](#). Change Log . . . . .

[26](#)     [A.1](#). 00 to 01 . . . . .

[26](#)     [A.2](#). 01 to 02 . . . . .

[26](#)     [A.3](#). 02 to 03 . . . . .

[26](#)     [A.4](#). 03 to 04 . . . . .

[26](#) Author's Address . . . . .

[27](#)

## 1. Introduction

This document defines a YANG [RFC7950] module for a system-level mechanism, herein called a "keystore". The keystore provides a centralized location for security sensitive data, as described below.

This module has the following characteristics:

- o A 'grouping' for a public/private key pair, and an 'action' for requesting the system to generate a new private key.
- o A 'grouping' for a list of certificates that might be associated with a public/private key pair, and an 'action' the requesting a system to generate a certificate signing request.
- o An unordered list of pinned certificate sets, where each pinned certificate set contains an unordered list of pinned certificates.  
This structure enables a server to use specific sets of pinned certificates on a case-by-case basis. For instance, one set of pinned certificates might be used by an HTTPS-client when connecting to particular HTTPS servers, while another set of pinned certificates might be used by a server when authenticating client connections (e.g., certificate-based client authentication).
- o An unordered list of pinned SSH host key sets, where each pinned SSH host key set contains an unordered list of pinned SSH host keys. This structure enables a server to use specific sets of pinned SSH host-keys on a case-by-case basis. For instance, SSH clients can be configured to use different sets of pinned SSH host keys when connecting to different SSH servers.
- o A notification to indicate when a certificate is about to expire.

Special consideration has been given for systems that have Trusted Protection Modules (TPMs). These systems are unique in that the TPM must be directed to generate new keys (it is not possible to load a key into a TPM) and it is not possible to backup/restore the TPM's private keys as configuration.

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

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in

Watsen  
3]

Expires May 3, 2018

[Page

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. Tree Diagram

The following tree diagram [I-D.ietf-netmod-yang-tree-diagrams] provides an overview of the data model for the "ietf-keystore" module.

```
module: ietf-keystore
  +--rw keystore
    +--rw pinned-certificates* [name]
      | +--rw name          string
      | +--rw description?  string
      | +--rw pinned-certificate* [name]
      |   +--rw name      string
      |   +--rw data      binary
    +--rw pinned-host-keys* [name]
      +--rw name          string
      +--rw description?  string
      +--rw pinned-host-key* [name]
        +--rw name      string
        +--rw data      binary

notifications:
  +---n certificate-expiration
    +--ro certificate      instance-identifier
    +--ro expiration-date yang:date-and-time

grouping certificate-grouping
  +---- certificates
  | +---- certificate* [name]
  |   +---- name?     string
  |   +---- value?    binary
  +---x generate-certificate-signing-request
    +---w input
    | +---w subject      binary
    | +---w attributes?  binary
    +--ro output
      +--ro certificate-signing-request  binary

grouping private-key-grouping
  +---- algorithm?      identityref
  +---- private-key?    union
  +---- public-key?     binary
  +---x generate-private-key
    +---w input
      +---w algorithm    identityref
```





### 3. Example Usage

The following example illustrates what a configured keystore might look like.

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

  <!-- Manufacturer's trust root CA certs -->
  <pinned-certificates>
    <name>manufacturers-root-ca-certs</name>
    <description>
      Certificates built into the device for authenticating
      manufacturer-signed objects, such as TLS server certificates,
      vouchers, etc.. Note, though listed here, these are not
      configurable; any attempt to do so will be denied.
    </description>
    <pinned-certificate>
      <name>Manufacturer Root CA cert 1</name>
      <data>base64encodedvalue==</data>
    </pinned-certificate>
    <pinned-certificate>
      <name>Manufacturer Root CA cert 2</name>
      <data>base64encodedvalue==</data>
    </pinned-certificate>
  </pinned-certificates>

  <!-- pinned netconf/restconf client certificates -->
  <pinned-certificates>
    <name>explicitly-trusted-client-certs</name>
    <description>
      Specific client authentication certificates for explicitly
      trusted clients. These are needed for client certificates
      that are not signed by a pinned CA.
    </description>
    <pinned-certificate>
      <name>George Jetson</name>
      <data>base64encodedvalue==</data>
    </pinned-certificate>
  </pinned-certificates>

  <!-- pinned netconf/restconf server certificates -->
  <pinned-certificates>
    <name>explicitly-trusted-server-certs</name>
    <description>
      Specific server authentication certificates for explicitly
      trusted servers. These are needed for server certificates
      that are not signed by a pinned CA.
    </description>
```



```
<pinned-certificate>
  <name>Fred Flintstone</name>
  <data>base64encodedvalue==</data>
</pinned-certificate>
</pinned-certificates>

<!-- trust anchors (CA certs) for authenticating clients -->
<pinned-certificates>
  <name>deployment-specific-ca-certs</name>
  <description>
    Trust anchors (i.e. CA certs) that are used to authenticate
    client connections. Clients are authenticated if their
    certificate has a chain of trust to one of these configured
    CA certificates.
  </description>
  <pinned-certificate>
    <name>ca.example.com</name>
    <data>base64encodedvalue==</data>
  </pinned-certificate>
</pinned-certificates>

<!-- trust anchors for random HTTPS servers on Internet -->
<pinned-certificates>
  <name>common-ca-certs</name>
  <description>
    Trusted certificates to authenticate common HTTPS servers.
    These certificates are similar to those that might be
    shipped with a web browser.
  </description>
  <pinned-certificate>
    <name>ex-certificate-authority</name>
    <data>base64encodedvalue==</data>
  </pinned-certificate>
</pinned-certificates>

<!-- pinned SSH host keys -->
<pinned-host-keys>
  <name>explicitly-trusted-ssh-host-keys</name>
  <description>
    Trusted SSH host keys used to authenticate SSH servers.
    These host keys would be analogous to those stored in
    a known_hosts file in OpenSSH.
  </description>
  <pinned-host-key>
    <name>corp-fw1</name>
    <data>base64encodedvalue==</data>
  </pinned-host-key>
</pinned-host-keys>
```



```
</keystore>
```

The following example illustrates the "certificate-expiration" notification in use with the NETCONF protocol.

[ note: '\ ' line wrapping for formatting only]

```
<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2016-07-08T00:01:00Z</eventTime>
  <certificate-expiration
    xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
    <certificate xmlns:ks="urn:ietf:params:xml:ns:yang:ietf-
keystore\
">
      /ks:keystore/ks:keys/ks:key[ks:name='ex-rsa-key']/
ks:certifica\
tes/ks:certificate[ks:name='ex-rsa-cert']
    </certificate>
    <expiration-date>2016-08-08T14:18:53-05:00</expiration-date>
    </certificate-expiration>
  </notification>
```

The following example module has been constructed to illustrate the groupings defined in the "ietf-keystore" module.

Watsen  
7]

Expires May 3, 2018

[Page

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

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

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

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

  contact
    "WG Web: <http://tools.ietf.org/wg/netconf/>
    WG List: <mailto:netconf@ietf.org>
    Author: Kent Watsen <mailto:kwatsen@juniper.net>";

  description
    "This module uses the groupings defines the keystore draft
    for illustration.";

  revision "YYYY-MM-DD" {
    description
      "Initial version";
  }

  container key {
    uses ks:private-key-grouping;
    uses ks:certificate-grouping;
    description
      "A container of certificates, and an action to generate
      a certificate signing request.";
  }
}
```

The following example illustrates what a configured key might look like. This example uses the "ex-keystore-usage" module above.





[ note: '\\' line wrapping for formatting only]

```
<key xmlns="http://example.com/ns/example-keystore-usage">
  <algorithm xmlns:ks="urn:ietf:params:xml:ns:yang:ietf-keystore">ks:
  \ secp521r1</algorithm>
  <private-key>base64encodedvalue==</private-key>
  <public-key>base64encodedvalue==</public-key>
  <certificates>
    <certificate>
      <name>domain certificate</name>
      <value>base64encodedvalue==</value>
    </certificate>
  </certificates>
</key>
```

The following example illustrates the "generate-certificate-signing-request" action in use with the NETCONF protocol. This example uses the "ex-keystore-usage" module above.

REQUEST

-----

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:
1.0">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <key xmlns="http://example.com/ns/example-keystore-usage">
      <generate-certificate-signing-request>
        <subject>base64encodedvalue==</subject>
        <attributes>base64encodedvalue==</attributes>
      </generate-certificate-signing-request>
    </key>
  </action>
</rpc>
```

RESPONSE

-----

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <certificate-signing-request
    xmlns="http://example.com/ns/example-keystore-usage">
    base64encodedvalue==
  </certificate-signing-request>
</rpc-reply>
```

The following example illustrates the "generate-private-key" action in use with the NETCONF protocol. This example uses the "ex-keystore-usage" module above.

Watsen  
9]

Expires May 3, 2018

[Page

REQUEST

-----

[ note: '\\' line wrapping for formatting only]

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:
1.0\
">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <key xmlns="http://example.com/ns/example-keystore-usage">
      <generate-private-key>
        <algorithm xmlns:ks="urn:ietf:params:xml:ns:yang:ietf-
keysto\
re">ks:secp521r1</algorithm>
          </generate-private-key>
        </key>
      </action>
    </rpc>
```

RESPONSE

-----

```
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

#### 4. YANG Module

This YANG module imports modules defined in [\[RFC6536\]](#) and [\[RFC6991\]](#). This module uses data types defined in [\[RFC2315\]](#), [\[RFC2986\]](#), [\[RFC3447\]](#), [\[RFC4253\]](#), [\[RFC5280\]](#), [\[RFC5915\]](#), and [\[ITU.X690.1994\]](#). This module uses algorithms defined in [\[RFC3447\]](#) and [\[RFC5480\]](#).

```
<CODE BEGINS> file "ietf-keystore@2017-10-30.yang"
module ietf-keystore {
  yang-version 1.1;

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

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 6536: Network Configuration Protocol (NETCONF) Access
      Control Model";
```

Watsen  
10]

Expires May 3, 2018

[Page

```
}
```

```
organization
```

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

```
contact
```

```
"WG Web: <http://tools.ietf.org/wg/netconf/>
```

```
WG List: <mailto:netconf@ietf.org>
```

```
Author: Kent Watsen
```

```
<mailto:kwatsen@juniper.net>";
```

```
description
```

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

```
Copyright (c) 2017 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  
(http://trustee.ietf.org/license-info).
```

```
This version of this YANG module is part of RFC VVVV; see  
the RFC itself for full legal notices.";
```

```
revision "2017-10-30" {
```

```
description
```

```
"Initial version";
```

```
reference
```

```
"RFC VVVV: YANG Data Model for a 'Keystore' Mechanism";
```

```
}
```

```
// Identities
```

```
identity key-algorithm {
```

```
description
```

```
"Base identity from which all key-algorithms are derived.";
```

```
}
```

```
identity rsa1024 {
```

```
base key-algorithm;
```

```
description
```

```
"The RSA algorithm using a 1024-bit key.";
```



```
reference
  "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
    RSA Cryptography Specifications Version 2.1.";
}

identity rsa2048 {
  base key-algorithm;
  description
    "The RSA algorithm using a 2048-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity rsa3072 {
  base key-algorithm;
  description
    "The RSA algorithm using a 3072-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity rsa4096 {
  base key-algorithm;
  description
    "The RSA algorithm using a 4096-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity rsa7680 {
  base key-algorithm;
  description
    "The RSA algorithm using a 7680-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
}

identity rsa15360 {
  base key-algorithm;
  description
    "The RSA algorithm using a 15360-bit key.";
  reference
    "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
      RSA Cryptography Specifications Version 2.1.";
```





```
    }

    identity secp192r1 {
      base key-algorithm;
      description
        "The secp192r1 algorithm.";
      reference
        "RFC5480:
        Elliptic Curve Cryptography Subject Public Key
Information.";
    }

    identity secp256r1 {
      base key-algorithm;
      description
        "The secp256r1 algorithm.";
      reference
        "RFC5480:
        Elliptic Curve Cryptography Subject Public Key
Information.";
    }

    identity secp384r1 {
      base key-algorithm;
      description
        "The secp384r1 algorithm.";
      reference
        "RFC5480:
        Elliptic Curve Cryptography Subject Public Key
Information.";
    }

    identity secp521r1 {
      base key-algorithm;
      description
        "The secp521r1 algorithm.";
      reference
        "RFC5480:
        Elliptic Curve Cryptography Subject Public Key
Information.";
    }

    // typedefs

    typedef pinned-certificates {
      type leafref {
        path "/ks:keystore/ks:pinned-certificates/ks:name";
      }
      description
        "This typedef enables importing modules to easily define a
reference to pinned-certificates. Use of this type also
```



```
        impacts the YANG tree diagram output.";
    reference
        "I-D.ietf-netmod-yang-tree-diagrams: YANG Tree Diagrams";
}

typedef pinned-host-keys {
    type leafref {
        path "/ks:keystore/ks:pinned-host-keys/ks:name";
    }
    description
        "This typedef enables importing modules to easily define a
        reference to pinned-host-keys. Use of this type also
        impacts the YANG tree diagram output.";
    reference
        "I-D.ietf-netmod-yang-tree-diagrams: YANG Tree Diagrams";
}

// groupings

grouping private-key-grouping {
    description
        "A private/public key pair, and an action to request the
        system to generate a private key.";
    leaf algorithm {
        type identityref {
            base "key-algorithm";
        }
        description
            "Identifies the key's algorithm. More specifically, this
            leaf specifies how the 'private-key' and 'public-key'
            binary leaves are encoded.";
    }
    leaf private-key {
        nacm:default-deny-all;
        type union {
            type binary;
            type enumeration {
                enum "hardware-protected" {
                    description
                        "The private key is inaccessible due to being
                        protected by a cryptographic hardware module
                        (e.g., a TPM).";
                }
            }
        }
    }
    must "../algorithm";
    description
```



```
    "A binary that contains the value of the private key. The
    interpretation of the content is defined by the key
    algorithm. For example, a DSA key is an integer, an RSA
    key is represented as RSAPrivateKey as defined in
    [RFC3447], and an Elliptic Curve Cryptography (ECC) key
    is represented as ECPrivateKey as defined in [RFC5915];
    reference
    "RFC 3447: Public-Key Cryptography Standards (PKCS) #1:
    RSA Cryptography Specifications Version 2.1.
    RFC 5915: Elliptic Curve Private Key Structure.";
}
leaf public-key {
    type binary;
    must "../algorithm";
    must "../private-key";
    description
    "A binary that contains the value of the public key. The
    interpretation of the content is defined by the key
    algorithm. For example, a DSA key is an integer, an RSA
    key is represented as RSAPublicKey as defined in
    [RFC3447], and an Elliptic Curve Cryptography (ECC) key
    is represented using the 'publicKey' described in
    [RFC5915];
    reference
    "RFC 3447: Public-Key Cryptography Standards (PKCS) #1:
    RSA Cryptography Specifications Version 2.1.
    RFC 5915: Elliptic Curve Private Key Structure.";
}
action generate-private-key {
    description
    "Requests the device to generate a private key using the
    specified key algorithm. This action is primarily to
    support cryptographic processors that must generate
    the private key themselves. The resulting key is
    considered operational state and hence only present
    in the <operational>.";
    input {
        leaf algorithm {
            type identityref {
                base "key-algorithm";
            }
            mandatory true;
            description
            "The algorithm to be used when generating the key.";
        }
    }
} // end generate-private-key
}
```



```
grouping certificate-grouping {
  description
    "A container of certificates, and an action to generate
    a certificate signing request.";
  container certificates {
    description
      "Certificates associated with this key. More than one
      certificate supports, for instance, a TPM-protected
      key that has both IDevID and LDevID certificates
      associated.";
    list certificate {
      key name;
      description
        "A certificate for this private key.";
      leaf name {
        type string;
        description
          "An arbitrary name for the certificate.";
      }
      leaf value {
        type binary;
        description
          "A PKCS #7 SignedData structure, as specified by
          Section 9.1 in RFC 2315, containing just certificates
          (no content, signatures, or CRLs), encoded using ASN.1
          distinguished encoding rules (DER), as specified in
          ITU-T X.690.

          This structure contains the certificate itself as well
          as any intermediate certificates leading up to a trust
          anchor certificate. The trust anchor certificate MAY
          be included as well.";
        reference
          "RFC 2315:
          PKCS #7: Cryptographic Message Syntax Version 1.5.
          ITU-T X.690:
          Information technology - ASN.1 encoding rules:
          Specification of Basic Encoding Rules (BER),
          Canonical Encoding Rules (CER) and Distinguished
          Encoding Rules (DER).";
      }
    }
  }
}
action generate-certificate-signing-request {
  description
    "Generates a certificate signing request structure for
    the associated private key using the passed subject and
    attribute values. The specified assertions need to be
```





```
appropriate for the certificate's use. For example,
an entity certificate for a TLS server SHOULD have
values that enable clients to satisfy RFC 6125
processing.";
input {
  leaf subject {
    type binary;
    mandatory true;
    description
      "The 'subject' field from the CertificationRequestInfo
      structure as specified by RFC 2986, Section 4.1 encoded
      using the ASN.1 distinguished encoding rules (DER), as
      specified in ITU-T X.690.";
    reference
      "RFC 2986:
      PKCS #10: Certification Request Syntax Specification
      Version 1.7.
      ITU-T X.690:
      Information technology - ASN.1 encoding rules:
      Specification of Basic Encoding Rules (BER),
      Canonical Encoding Rules (CER) and Distinguished
      Encoding Rules (DER).";
  }
  leaf attributes {
    type binary;
    description
      "The 'attributes' field from the CertificationRequestInfo
      structure as specified by RFC 2986, Section 4.1 encoded
      using the ASN.1 distinguished encoding rules (DER), as
      specified in ITU-T X.690.";
    reference
      "RFC 2986:
      PKCS #10: Certification Request Syntax Specification
      Version 1.7.
      ITU-T X.690:
      Information technology - ASN.1 encoding rules:
      Specification of Basic Encoding Rules (BER),
      Canonical Encoding Rules (CER) and Distinguished
      Encoding Rules (DER).";
  }
}
output {
  leaf certificate-signing-request {
    type binary;
    mandatory true;
    description
      "A CertificationRequest structure as specified by RFC
      2986, Section 4.1 encoded using the ASN.1 distinguished
```



```
        encoding rules (DER), as specified in ITU-T X.690.";
    reference
        "RFC 2986:
        PKCS #10: Certification Request Syntax Specification
        Version 1.7.
        ITU-T X.690:
        Information technology - ASN.1 encoding rules:
        Specification of Basic Encoding Rules (BER),
        Canonical Encoding Rules (CER) and Distinguished
        Encoding Rules (DER).";
    }
}
}
}

// protocol accessible nodes

container keystore {
    nacm:default-deny-write;
    description
        "The keystore contains X.509 certificates and SSH host keys.";

    list pinned-certificates {
        key name;
        description
            "A list of pinned certificates. These certificates can be
            used by a server to authenticate clients, or by clients to
            authenticate servers. Each list of pinned certificates
            SHOULD be specific to a purpose, as the list as a whole
            may be referenced by other modules. For instance, a
            NETCONF server's configuration might use a specific list
            of pinned certificates for when authenticating NETCONF
            client connections.";
        leaf name {
            type string;
            description
                "An arbitrary name for this list of pinned certificates.";
        }
        leaf description {
            type string;
            description
                "An arbitrary description for this list of pinned
                certificates.";
        }
    }
    list pinned-certificate {
        key name;
```



```
description
  "A pinned certificate.";
leaf name {
  type string;
  description
    "An arbitrary name for this pinned certificate. The
    name must be unique across all lists of pinned
    certificates (not just this list) so that leafrefs
    from another module can resolve to unique values.";
}
leaf data {
  type binary;
  mandatory true;
  description
    "An X.509 v3 certificate structure as specified by RFC
    5280, Section 4 encoded using the ASN.1 distinguished
    encoding rules (DER), as specified in ITU-T X.690.";
  reference
    "RFC 5280:
    Internet X.509 Public Key Infrastructure Certificate
    and Certificate Revocation List (CRL) Profile.
    ITU-T X.690:
    Information technology - ASN.1 encoding rules:
    Specification of Basic Encoding Rules (BER),
    Canonical Encoding Rules (CER) and Distinguished
    Encoding Rules (DER).";
}
}
}

list pinned-host-keys {
  key name;
  description
    "A list of pinned host keys. These pinned host-keys can
    be used by clients to authenticate SSH servers. Each
    list of pinned host keys SHOULD be specific to a purpose,
    so the list as a whole may be referenced by other modules.
    For instance, a NETCONF client's configuration might
    point to a specific list of pinned host keys for when
    authenticating specific SSH servers.";
  leaf name {
    type string;
    description
      "An arbitrary name for this list of pinned SSH host keys.";
  }
  leaf description {
    type string;
    description
```



```
        "An arbitrary description for this list of pinned SSH host
        keys.";
    }
    list pinned-host-key {
        key name;
        description
            "A pinned host key.";
        leaf name {
            type string;
            description
                "An arbitrary name for this pinned host-key. Must be
                unique across all lists of pinned host-keys (not just
                this list) so that a leafref to it from another module
                can resolve to unique values.";
        }
        leaf data {
            type binary;
            mandatory true;
            description
                "The binary public key data for this SSH key, as
                specified by RFC 4253, Section 6.6, i.e.:

                string    certificate or public key format
                        identifier
                byte[n]   key/certificate data.";
            reference
                "RFC 4253: The Secure Shell (SSH) Transport Layer
                Protocol";
        }
    }
}

notification certificate-expiration {
    description
        "A notification indicating that a configured certificate is
        either about to expire or has already expired. When to send
        notifications is an implementation specific decision, but
        it is RECOMMENDED that a notification be sent once a month
        for 3 months, then once a week for four weeks, and then once
        a day thereafter.";
    leaf certificate {
        type instance-identifier;
        mandatory true;
        description
            "Identifies which certificate is expiring or is expired.";
    }
    leaf expiration-date {
```





```
    type yang:date-and-time;
    mandatory true;
    description
      "Identifies the expiration date on the certificate.";
  }
}
}
<CODE ENDS>
```

## 5. Security Considerations

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

The NETCONF access control model (NACM) [[RFC6536](#)] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/: The entire data tree defined by this module is sensitive to write operations. For instance, the addition or removal of keys, certificates, trusted anchors, etc., can dramatically alter the implemented security policy. This being the case, the top-level node in this module is marked with the NACM value 'default-deny-write'.

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

this node, implementations SHOULD automatically (without explicit request) zeroize these keys in the most secure manner

Watsen  
21]

Expires May 3, 2018

[Page

available, so as to prevent the remnants of their persisted storage locations from being analyzed in any meaningful way.

Some of the readable data nodes in this YANG module may be considered

sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

it /keystore/keys/key/private-key: This node is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. The best reason for returning this node is to support backup/restore type workflows. This being the case, this node is marked with the NACM value 'default-deny-all'.

Some of the operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

sent generate-certificate-signing-request: For this action, it is RECOMMENDED that implementations assert channel binding [[RFC5056](#)], so as to ensure that the application layer that sent the request is the same as the device authenticated when the secure transport layer was established.

## **6. IANA Considerations**

### **6.1. The IETF XML Registry**

This document registers one URI in 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 the following registration is requested:

Watsen  
22]

Expires May 3, 2018

[Page

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

## 7. Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin

Bjorklund, Benoit Claise, Mehmet Ersue, Balazs Kovacs, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, Juergen Schoenwaelder; Phil Shafer, Sean Turner, and Bert Wijnen.

## 8. References

### 8.1. Normative References

[ITU.X690.1994]

International Telecommunications Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 1994.

[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>>.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", [RFC 2315](#), DOI 10.17487/RFC2315, March 1998, <<https://www.rfc-editor.org/info/rfc2315>>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", [RFC 2986](#), DOI 10.17487/RFC2986, November 2000, <<https://www.rfc-editor.org/info/rfc2986>>.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", [RFC 3447](#), DOI 10.17487/RFC3447, February 2003, <<https://www.rfc-editor.org/info/rfc3447>>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", [RFC 4253](#), DOI 10.17487/RFC4253, January 2006, <<https://www.rfc-editor.org/info/rfc4253>>.

Watsen  
23]

Expires May 3, 2018

[Page

- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 5280](#), DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", [RFC 5480](#), DOI 10.17487/RFC5480, March 2009, <<https://www.rfc-editor.org/info/rfc5480>>.
- [RFC5915] Turner, S. and D. Brown, "Elliptic Curve Private Key Structure", [RFC 5915](#), DOI 10.17487/RFC5915, June 2010, <<https://www.rfc-editor.org/info/rfc5915>>.
- [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>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", [RFC 6536](#), DOI 10.17487/RFC6536, March 2012, <<https://www.rfc-editor.org/info/rfc6536>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", [RFC 6991](#), DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [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>>.

## **8.2. Informative References**

- [I-D.ietf-netmod-yang-tree-diagrams] Bjorklund, M. and L. Berger, "YANG Tree Diagrams", [draft-ietf-netmod-yang-tree-diagrams-02](#) (work in progress), October 2017.
- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", [RFC 4211](#), DOI 10.17487/RFC4211, September 2005, <<https://www.rfc-editor.org/info/rfc4211>>.

Watsen  
24]

Expires May 3, 2018

[Page



- [RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", [RFC 5056](#), DOI 10.17487/RFC5056, November 2007, <<https://www.rfc-editor.org/info/rfc5056>>.
- [RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Format", [RFC 5914](#), DOI 10.17487/RFC5914, June 2010, <<https://www.rfc-editor.org/info/rfc5914>>.
- [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>>.
- [Std-802.1AR-2009] IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.



## [Appendix A.](#) Change Log

### [A.1.](#) 00 to 01

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

### [A.2.](#) 01 to 02

- o Added back 'generate-private-key' action.
- o Removed 'RESTRICTED' enum from the 'private-key' leaf type.
- o Fixed up a few description statements.

### [A.3.](#) 02 to 03

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

### [A.4.](#) 03 to 04

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



Internet-Draft YANG Data Model for a "Keystore" Mechanism   October  
2017

Author's Address

Kent Watsen  
Juniper Networks

E-Mail: [kwatsen@juniper.net](mailto:kwatsen@juniper.net)

