NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 23, 2020

K. Watsen
Watsen Networks
November 20, 2019

A YANG Data Model for a Truststore
draft-ietf-netconf-trust-anchors-08

Abstract

This document defines a YANG 1.1 data model for configuring global sets of X.509 certificates, SSH host-keys, raw public keys, and PSKs (pairwise-symmetric or pre-shared keys) that can be referenced by other data models for trust. While the SSH host-keys are uniquely for the SSH protocol, certificates, raw public keys, and PSKs may have multiple uses, including authenticating protocol peers and verifying signatures.

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

o "YYYY" --> the assigned RFC value for draft-ietf-netconf-crypto-types

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

o "2019-11-20" --> 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

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG 1.1 [RFC7950] data model for configuring global sets of X.509 certificates, SSH host-keys, raw public keys, and PSKs (pairwise-symmetric or pre-shared keys) that can be referenced by other data models for trust. While the SSH host-keys are uniquely for the SSH protocol, certificates, raw public keys, and PSKs may have multiple uses, including authenticating protocol peers and verifying signatures.

This document in compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, trust anchors installed during manufacturing (e.g., for trusted well-known services), are expected to appear in <operational> (see Section 3).

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

1.2. Tree Diagram Notation

Tree diagrams used in this document follow the notation defined in [RFC8340].

2. The Trust Anchors Model

2.1. Tree Diagram

The following tree diagram provides an overview of the "ietf-truststore" module.

```
module: ietf-truststore
  +--rw truststore
     +--rw certificates* [name] {x509-certificates}?
     |  +--rw name           string
     |  +--rw description?   string
```

```
        |  +--rw certificate* [name]
        |     +--rw name                      string
        |     +--rw cert                      trust-anchor-cert-cms
        |     +---n certificate-expiration
        |        +-- expiration-date    yang:date-and-time
        +--rw host-keys* [name] {ssh-host-keys}?
        |  +--rw name           string
        |  +--rw description?   string
        |  +--rw host-key* [name]
        |     +--rw name        string
        |     +--rw host-key    ct:ssh-host-key
        +--rw raw-public-keys* [name] {raw-public-keys}?
           +--rw name              string
           +--rw description?      string
           +--rw raw-public-key* [name]
              +--rw name                 string
              +--rw algorithm            iasa:asymmetric-algorithm-type
              +--rw public-key-format?   identityref
              +--rw public-key           binary

     grouping local-or-truststore-certs-grouping
       +-- (local-or-truststore)
          +--:(local) {local-definitions-supported}?
          |  +-- local-definition
          |     +-- cert*                     trust-anchor-cert-cms
          |     +---n certificate-expiration
          |        +-- expiration-date    yang:date-and-time
          +--:(truststore) {truststore-supported,x509-certificates}?
             +-- truststore-reference?   ts:certificates-ref
     grouping local-or-truststore-host-keys-grouping
       +-- (local-or-truststore)
          +--:(local) {local-definitions-supported}?
          |  +-- local-definition
          |     +-- host-key*   ct:ssh-host-key
          +--:(truststore) {truststore-supported,ssh-host-keys}?
             +-- truststore-reference?   ts:host-keys-ref
     grouping local-or-truststore-raw-pub-keys-grouping
       +-- (local-or-truststore)
          +--:(local) {local-definitions-supported}?
          |  +-- local-definition
          |     +-- raw-public-key* [name]
          |        +-- name?                 string
          |        +-- algorithm
          |        |       iasa:asymmetric-algorithm-type
          |        +-- public-key-format?   identityref
          |        +-- public-key           binary
          +--:(truststore) {truststore-supported,raw-public-keys}?
             +-- truststore-reference?   ts:raw-public-keys-ref
```

```
     grouping truststore-grouping
       +-- certificates* [name] {x509-certificates}?
       |  +-- name?          string
       |  +-- description?   string
       |  +-- certificate* [name]
       |     +-- name?                     string
       |     +-- cert                      trust-anchor-cert-cms
       |     +---n certificate-expiration
       |        +-- expiration-date    yang:date-and-time
       +-- host-keys* [name] {ssh-host-keys}?
       |  +-- name?          string
       |  +-- description?   string
       |  +-- host-key* [name]
       |     +-- name?       string
       |     +-- host-key    ct:ssh-host-key
       +-- raw-public-keys* [name] {raw-public-keys}?
          +-- name?             string
          +-- description?      string
          +-- raw-public-key* [name]
             +-- name?                 string
             +-- algorithm             iasa:asymmetric-algorithm-type
             +-- public-key-format?   identityref
             +-- public-key            binary
```

2.2. Example Usage

The following example illustrates trust anchors in <intended>. Please see Section 3 for an example illustrating built-in values in <operational>.

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

     <!-- Manufacturer's trusted root CA certs -->
     <certificates or:origin="or:system">
       <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>
       <certificate>
         <name>Manufacturer Root CA cert 1</name>
         <cert>base64encodedvalue==</cert>
       </certificate>
       <certificate>
```

```
         <name>Manufacturer Root CA cert 2</name>
         <cert>base64encodedvalue==</cert>
       </certificate>
     </certificates>

     <!-- specific end-entity certs for authenticating servers -->
     <certificates or:origin="or:intended">
       <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 CA.
       </description>
       <certificate>
         <name>Fred Flintstone</name>
         <cert>base64encodedvalue==</cert>
       </certificate>
     </certificates>

     <!-- trusted CA certs for authenticating servers -->
     <certificates or:origin="or:intended">
       <name>explicitly-trusted-server-ca-certs</name>
       <description>
         Trust anchors (i.e. CA certs) that are used to authenticate
         server connections.  Servers are authenticated if their
         certificate has a chain of trust to one of these CA
         certificates.
       </description>
       <certificate>
         <name>ca.example.com</name>
         <cert>base64encodedvalue==</cert>
       </certificate>
     </certificates>

     <!-- specific end-entity certs for authenticating clients -->
     <certificates or:origin="or:intended">
       <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 CA.
       </description>
       <certificate>
         <name>George Jetson</name>
         <cert>base64encodedvalue==</cert>
       </certificate>
     </certificates>
```

```
     <!-- trusted CA certs for authenticating clients -->
     <certificates or:origin="or:intended">
       <name>explicitly-trusted-client-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 CA
         certificates.
       </description>
       <certificate>
         <name>ca.example.com</name>
         <cert>base64encodedvalue==</cert>
       </certificate>
     </certificates>

     <!-- trusted CA certs for random HTTPS servers on Internet -->
     <certificates or:origin="or:system">
       <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>
       <certificate>
         <name>ex-certificate-authority</name>
         <cert>base64encodedvalue==</cert>
       </certificate>
     </certificates>

     <!-- specific SSH host keys for authenticating clients -->
     <host-keys or:origin="or:intended">
       <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>
       <host-key>
         <name>corp-fw1</name>
         <host-key>base64encodedvalue==</host-key>
       </host-key>
     </host-keys>

   </truststore>
```

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

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

<notification
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <truststore xmlns="urn:ietf:params:xml:ns:yang:ietf-truststore">
    <certificates>
      <name>explicitly-trusted-client-certs</name>
      <certificate>
        <name>George Jetson</name>
        <certificate-expiration>
          <expiration-date>2018-08-05T14:18:53-05:00</expiration-dat\
e>
        </certificate-expiration>
      </certificate>
    </certificates>
  </truststore>
</notification>
```

2.3. YANG Module

This YANG module imports modules from [RFC8341] and [I-D.ietf-netconf-crypto-types].

```
<CODE BEGINS> file "ietf-truststore@2019-11-20.yang"

module ietf-truststore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-truststore";
  prefix ts;

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

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

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

  contact
```

```
       "WG Web  : <http://datatracker.ietf.org/wg/netconf/>
        WG List : <mailto:netconf@ietf.org>
        Author  : Kent Watsen <kent+ietf@watsen.net>
        Author  : Henk Birkholz <henk.birkholz@sit.fraunhofer.de>";

     description
       "This module defines a truststore to centralize management
        of trust anchors including X.509 certificates, SSH host
        keys, raw public keys, and PSKs (pairwise-symmetric or
        pre-shared keys).

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

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Simplified
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

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

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
        are to be interpreted as described in BCP 14 (RFC 2119)
        (RFC 8174) when, and only when, they appear in all
        capitals, as shown here.";

     revision 2019-11-20 {
       description
         "Initial version";
       reference
         "RFC XXXX: A YANG Data Model for a Truststore";
     }

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

     feature truststore-supported {
       description
         "The 'truststore-supported' feature indicates that the
          server supports the truststore (i.e., implements the
          'ietf-truststore' module).";
```

```
     }

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

     feature x509-certificates {
       description
         "The 'x509-certificates' feature indicates that the server
          implements the /truststore/certificates subtree.";
     }

     feature ssh-host-keys {
       description
         "The 'ssh-host-keys' feature indicates that the server
          implements the /truststore/host-keys subtree.";
     }

     feature raw-public-keys {
       description
         "The 'raw-public-keys' feature indicates that the server
          implements the /truststore/raw-public-keys subtree.";
     }

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

     typedef certificates-ref {
       type leafref {
         path "/ts:truststore/ts:certificates/ts:name";
       }
       description
         "This typedef enables modules to easily define a reference
          to a set of certificates  defined in the truststore.";
     }

     typedef host-keys-ref {
       type leafref {
         path "/ts:truststore/ts:host-keys/ts:name";
       }
       description
         "This typedef enables modules to easily define a reference
          to a set of host keys defined in the truststore.";
     }
```

```
     typedef raw-public-keys-ref {
       type leafref {
         path "/ts:truststore/ts:raw-public-keys/ts:name";
       }
       description
         "This typedef enables modules to easily define a reference
          to a set of raw public keys defined in the truststore.";
     }

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

     grouping local-or-truststore-certs-grouping {
       description
         "A grouping that expands to allow trust anchors to be
          either stored locally, within the using data model, or be
          a reference to trust anchors stored in the truststore.";
       choice local-or-truststore {
         mandatory true;
         case local {
           if-feature "local-definitions-supported";
           container local-definition {
             description
               "Container to hold the local trust anchor definitions.";
             uses ct:trust-anchor-certs-grouping;
           }
         }
         case truststore {
           if-feature "truststore-supported";
           if-feature "x509-certificates";
           leaf truststore-reference {
             type ts:certificates-ref;
             description
               "A reference to a set of trust anchors that exists
                in the truststore.";
           }
         }
         description
           "A choice between an inlined definition and a definition
            that exists in the truststore.";
       }
     }

     grouping local-or-truststore-host-keys-grouping {
       description
         "A grouping that expands to allow host keys to be
          either stored locally, within the using data model, or be
```

```
           a reference to host keys stored in the truststore.";
        choice local-or-truststore {
          mandatory true;
          case local {
            if-feature "local-definitions-supported";
            container local-definition {
              description
                "Container to hold local host key definitions.";
              leaf-list host-key {
                nacm:default-deny-write;
                type ct:ssh-host-key;
                description
                  "The binary public key data for this host key.";
                reference
                  "RFC YYYY: Common YANG Data Types for Cryptography";
              }
            }
          }
          case truststore {
            if-feature "truststore-supported";
            if-feature "ssh-host-keys";
            leaf truststore-reference {
              type ts:host-keys-ref;
              description
                "A reference to a set of host keys that exist in
                 the truststore.";
            }
          }
          description
            "A choice between an inlined definition and a definition
             that exists in the truststore.";
        }
      }

      grouping local-or-truststore-raw-pub-keys-grouping {
        description
          "A grouping that expands to allow raw public keys to be
           available either locally, within the using data model, or
           be a reference to raw public keys stored in the truststore.";
        choice local-or-truststore {
          mandatory true;
          case local {
            if-feature "local-definitions-supported";
            container local-definition {
              description
                "Container to hold local raw public key definitions.";
              list raw-public-key {
                key name;
```

```
               description
                 "A raw public key definition.";
               leaf name {
                 type string;
                 description
                   "An arbitrary name for this raw public key.";
               }
               uses ct:public-key-grouping;
             }
           }
         }
         case truststore {
           if-feature "truststore-supported";
           if-feature "raw-public-keys";
           leaf truststore-reference {
             type ts:raw-public-keys-ref;
             description
               "A reference to a set of raw public keys that exist
                in the truststore.";
           }
         }
         description
           "A choice between an inlined definition and a definition
            that exists in the truststore.";
       }
     }

     grouping truststore-grouping {
       description
         "Grouping definition enables use in other contexts.  If ever
          done, implementations SHOULD augment new 'case' statements
          into local-or-keystore 'choice' statements to supply leafrefs
          to the new location.";
       list certificates {
         if-feature "x509-certificates";
         key "name";
         description
           "A list of certificates.  These certificates can be
            used by a server to authenticate clients, or by a client
            to authenticate servers.  Each list of certificates
            SHOULD be specific to a purpose, as the list as a whole
            may be referenced by other modules.  For instance, a
            RESTCONF server's configuration might use a specific list
            of certificates for when authenticating RESTCONF
            client connections.";
         leaf name {
           type string;
           description
```

```
             "An arbitrary name for this list of certificates.";
         }
         leaf description {
           type string;
           description
             "An arbitrary description for this list of
              certificates.";
         }
         list certificate {
           key "name";
           description
             "A certificate.";
           leaf name {
             type string;
             description
               "An arbitrary name for this certificate. The
                name must be unique across all lists of
                certificates (not just this list) so that leafrefs
                from another module can resolve to unique values.";
           }
           uses ct:trust-anchor-cert-grouping {
             refine "cert" {
               mandatory true;
             }
           }
         }
       }
       list host-keys {
         if-feature "ssh-host-keys";
         key "name";
         description
           "A list of host keys.  These host-keys can be used by
            clients to authenticate SSH servers.  Each list of host
            keys SHOULD be specific to a purpose, so the list as a
            whole may be referenced by other modules.  For instance,
            a client's configuration might point to a specific list
            of host keys for when authenticating specific SSH servers.";
         leaf name {
           type string;
           description
             "An arbitrary name for this list of SSH
              host keys.";
         }
         leaf description {
           type string;
           description
             "An arbitrary description for this list of SSH
              host keys.";
```

```
         }
         list host-key {
           key "name";
           description
             "A host key.";
           leaf name {
             type string;
             description
               "An arbitrary name for this host-key.";
           }
           leaf host-key {
             type ct:ssh-host-key;
             mandatory true;
             description
               "The binary public key data for this host key.";
             reference
               "RFC YYYY: Common YANG Data Types for Cryptography";
           }
         }
       }
       list raw-public-keys {
         if-feature "raw-public-keys";
         key "name";
         description
           "A list of raw public keys. These raw public keys can be
            used by a server to authenticate clients, or by a client
            to authenticate servers. Each list of raw public keys
            SHOULD be specific to a purpose, so the list as a whole
            may be referenced by other modules.  For instance, a
            client's configuration might point to a specific list
            of raw public keys for when authenticating specific TLS
            endpoints.";
         leaf name {
           type string;
           description
             "An arbitrary name for this list of raw public keys.";
         }
         leaf description {
           type string;
           description
             "An arbitrary description for this list raw public keys.";
         }
         list raw-public-key {
           key "name";
           description
             "A raw public key.";
           leaf name {
             type string;
```

```
             description
               "An arbitrary name for this raw public key.";
           }
           uses ct:public-key-grouping;
         }
       }
     }

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

     container truststore {
       nacm:default-deny-write;
       description
         "The truststore contains sets of X.509 certificates and
          SSH host keys.";
       uses truststore-grouping;
     }
   }

   <CODE ENDS>
```

3. Support for Built-in Trust Anchors

In some implementations, the operating system a device is running, may define some built-in trust anchors. For instance, there may be built-in trust anchors enabling the device to securely connect to well-known services (e.g., an SZTP [RFC8572] bootstap server) or trust anchors to connect to arbitrary services using public PKI.

Built-in trust anchors are expected to be set by a vendor-specific process. Any ability for operators to modify built-in trust anchors is outside the scope of this docuemnt.

As built-in trust anchors are provided by the system (not configuration), they are present in <operational>. The following example illustrates bui;t-in trust anchors in <operational>.

(FIXME: add illustration with origin="system" here)

In order for the built-in trust anchors to be referenced by configuration, they must first be copied into <intended> as the example in Section 2.2 illustrates for the built-in trust anchors above. Note that this strategy is chosen, rather then setting "require-instance false" for the various leafrefs, as built-in trust anchors are relatively few in number and hence not worth relaxing the validation for.

4. Security Considerations

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

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

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

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

None of the readable data nodes in this YANG module are considered sensitive or vulnerable in network environments.

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

5. IANA Considerations

5.1. The IETF XML Registry

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

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

5.2. The YANG Module Names Registry

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

```
   name:         ietf-truststore
   namespace:    urn:ietf:params:xml:ns:yang:ietf-truststore
   prefix:       ta
   reference:    RFC XXXX
```

6. References

6.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-12 (work in progress), November 2019.

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

[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>.

[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>.

6.2. Informative References

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

[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>.

[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>.

[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>.

[RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <https://www.rfc-editor.org/info/rfc8572>.

Appendix A. Change Log

A.1. 00 to 01

- Added features "x509-certificates" and "ssh-host-keys".

- Added nacm:default-deny-write to "trust-anchors" container.

A.2. 01 to 02

- Switched "list pinned-certificate" to use the "trust-anchor-cert-grouping" from crypto-types. Effectively the same definition as before.

A.3. 02 to 03

- Updated copyright date, boilerplate template, affiliation, folding algorithm, and reformatted the YANG module.

A.4. 03 to 04

- Added groupings 'local-or-truststore-certs-grouping' and 'local-or-truststore-host-keys-grouping', matching similar definitions in the keystore draft. Note new (and incomplete) "truststore" usage!

- Related to above, also added features 'truststore-supported' and 'local-trust-anchors-supported'.

A.5. 04 to 05

- Renamed "trust-anchors" to "truststore"

- Removed "pinned." prefix everywhere, to match truststore rename

- Moved everything under a top-level 'grouping' to enable use in other contexts.

- Renamed feature from 'local-trust-anchors-supported' to 'local-definitions-supported' (same name used in keystore)

- Removed the "require-instance false" statement from the "*-ref" typedefs.

- Added missing "ssh-host-keys" and "x509-certificates" if-feature statements

A.6. 05 to 06

- Editorial changes only.

A.7. 06 to 07

- Added Henk Birkholz as a co-author (thanks Henk!)

- Added PSKs and raw public keys to Truststore.

A.8. 07 to 08

- Added new "Support for Built-in Trust Anchors" section.

- Removed spurious "uses ct:trust-anchor-certs-grouping" line.

Acknowledgements

The authors especially thank Henk Birkholz for contributing YANG to the ietf-truststore module supporting raw public keys and PSKs (pre-shared or pairwise-symmetric keys). While these contributions were eventually replaced by reusing the existing support for asymmetric and symmetric trust anchors, respectively, it was only thru Henk's initiative that the WG was able to come to that result.

The authors additionally thank the following for helping give shape to this work (ordered by last name): Martin Bjorklund, Nick Hancock, Balazs Kovacs, Eric Voit, and Liang Xia.

Author's Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net