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

K. Watsen
Juniper Networks
G. Wu
Cisco Systems

October 30, 2017

YANG Groupings for SSH Clients and SSH Servers
draft-ietf-netconf-ssh-client-server-04

Abstract

This document defines three YANG modules: the first defines groupings for a generic SSH client, the second defines groupings for a generic SSH server, and the third defines common identities and groupings used by both the client and the server. It is intended that these groupings will be used by applications using the SSH protocol.

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.

This document contains references to other drafts in progress, both in the Normative References section, as well as in body text throughout. Please update the following references to reflect their final RFC assignments:

o I-D.ietf-netconf-keystore

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 I-D.ietf-netconf-keystore

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
2. Terminology 3
3. The SSH Client Model 4
  3.1. Tree Diagram 4
  3.2. Example Usage 5
  3.3. YANG Module 6
4. The SSH Server Model 10
  4.1. Tree Diagram 10
  4.2. Example Usage 12
  4.3. YANG Module 13
5. The SSH Common Model 16
  5.1. Tree Diagram 17
  5.2. Example Usage 17
  5.3. YANG Module 18

6. Security Considerations 28
7. IANA Considerations 29
7.1. The IETF XML Registry 29
7.2. The YANG Module Names Registry 30
8. Acknowledgements 30
9. References 30
9.1. Normative References 31
9.2. Informative References 32
Appendix A. Change Log 33
A.1. 00 to 01 33
A.2. 01 to 02 33
A.3. 02 to 03 33
A.4. 03 to 04 33
Authors' Addresses 34

1. Introduction

This document defines three YANG [RFC7950] modules: the first defines a grouping for a generic SSH client, the second defines a grouping for a generic SSH server, and the third defines identities and groupings common to both the client and the server (SSH is defined in [RFC4252], [RFC4253], and [RFC4254]). It is intended that these groupings will be used by applications using the SSH protocol. For instance, these groupings could be used to help define the data model for an OpenSSH [OPENSSH] server or a NETCONF over SSH [RFC6242] based server.

The client and server YANG modules in this document each define one grouping, which is focused on just SSH-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen-on or connect-to. This enables applications the opportunity to define their own strategy for how the underlying TCP connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the grouping for the SSH parts it provides, while adding data nodes for the TCP-level call-home configuration.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. The SSH Client Model

The SSH client model presented in this section contains one YANG grouping, to just configure the SSH client, omitting, for instance, any configuration for which IP address or port the client should connect to.

This grouping references data nodes defined by the keystore model [I-D.ietf-netconf-keystore]. For instance, a reference to the keystore model is made to indicate which trusted CA certificate a client should use to authenticate X.509v3 certificate based host keys [RFC6187].

3.1. Tree Diagram

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

```
module: ietf-ssh-client

  grouping ssh-client-grouping
    +---- client-identity
    |  +---- username?      string
    |  +---- (auth-type)
    |     +--:(certificate)
    |     |  +---- certificate {sshcom:ssh-x509-certs}?
    |     |     +---- algorithm?
    |     |     |       identityref
    |     |     +---- private-key?                          union
    |     |     +---- public-key?                           binary
    |     |     +---x generate-private-key
    |     |     |  +---w input
    |     |     |     +---w algorithm    identityref
    |     |     +---- 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
    |     +--:(public-key)
    |     |  +---- public-key
    |     |     +---- algorithm?              identityref
    |     |     +---- private-key?            union
```

```
      |     |     +---- public-key?             binary
      |     |     +---x generate-private-key
      |     |        +---w input
      |     |           +---w algorithm    identityref
      |     +--:(password)
      |        +---- password?      string
      +---- server-auth
      |  +---- pinned-ssh-host-keys?   ks:pinned-host-keys
      |  +---- pinned-ca-certs?        ks:pinned-certificates
      |  |       {sshcom:ssh-x509-certs}?
      |  +---- pinned-server-certs?    ks:pinned-certificates
      |          {sshcom:ssh-x509-certs}?
      +---- transport-params {ssh-client-transport-params-config}?
         +---- host-key
         |  +---- host-key-alg*   identityref
         +---- key-exchange
         |  +---- key-exchange-alg*   identityref
         +---- encryption
         |  +---- encryption-alg*   identityref
         +---- mac
         |  +---- mac-alg*   identityref
         +---- compression
            +---- compression-alg*   identityref
```

3.2. Example Usage

This section shows how it would appear if the ssh-client-grouping were populated with some data. This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-keystore].

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

<!-- hypothetical example, as groupings don't have instance data -->\

<ssh-client xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-client"
            xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common"\
>

  <!-- how this client will authenticate itself to the server -->
  <client-identity>
    <username>foobar</username>
    <public-key>
      <algorithm xmlns:ks="urn:ietf:params:xml:ns:yang:ietf-keystore\
">ks:secp521r1</algorithm>
      <private-key>base64encodedvalue==</private-key>
      <public-key>base64encodedvalue==</public-key>
    </public-key>
  </client-identity>
```

```
     <!-- which host-keys will this client trust -->
     <server-auth>
       <pinned-ssh-host-keys>explicitly-trusted-ssh-host-keys</pinned-s\
   sh-host-keys>
     </server-auth>

     <transport-params>
       <host-key>
         <host-key-alg>algs:ssh-rsa</host-key-alg>
       </host-key>
       <key-exchange>
         <key-exchange-alg>
           algs:diffie-hellman-group-exchange-sha256
         </key-exchange-alg>
       </key-exchange>
       <encryption>
         <encryption-alg>algs:aes256-ctr</encryption-alg>
         <encryption-alg>algs:aes192-ctr</encryption-alg>
         <encryption-alg>algs:aes128-ctr</encryption-alg>
         <encryption-alg>algs:aes256-cbc</encryption-alg>
         <encryption-alg>algs:aes192-cbc</encryption-alg>
         <encryption-alg>algs:aes128-cbc</encryption-alg>
       </encryption>
       <mac>
         <mac-alg>algs:hmac-sha2-256</mac-alg>
         <mac-alg>algs:hmac-sha2-512</mac-alg>
       </mac>
       <compression>
         <compression-alg>algs:none</compression-alg>
       </compression>

     </transport-params>

   </ssh-client>
```

3.3. YANG Module

This YANG module has a normative references to [RFC6991] and [I-D.ietf-netconf-keystore].

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

  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-client";
  prefix "sshc";

  import ietf-ssh-common {
```

```
      prefix sshcom;
      revision-date 2017-10-30; // stable grouping definitions
      reference
        "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
    }

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

    import ietf-keystore {
      prefix ks;
      reference
        "RFC YYYY: Keystore Model";
    }

    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>

      Author:   Gary Wu
                <mailto:garywu@cisco.com>";


    description
     "This module defines a reusable grouping for a SSH client that
      can be used as a basis for specific SSH client instances.

      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 XXXX; see
```

```
       the RFC itself for full legal notices.";

     revision "2017-10-30" {
       description
        "Initial version";
       reference
        "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
     }

     // features

     feature ssh-client-transport-params-config {
       description
         "SSH transport layer parameters are configurable on an SSH
          client.";
     }

     // groupings

     grouping ssh-client-grouping {
       description
         "A reusable grouping for configuring a SSH client without
          any consideration for how an underlying TCP session is
          established.";

       container client-identity {
         description
           "The credentials used by the client to authenticate to
            the SSH server.";

         leaf username {
           type string;
           description
             "The username of this user.  This will be the username
              used, for instance, to log into an SSH server.";
         }

         choice auth-type {
           mandatory true;
           description
             "The authentication type.";
           container certificate {
             if-feature sshcom:ssh-x509-certs;
             uses ks:private-key-grouping;
             uses ks:certificate-grouping;
             description
               "A certificates to be used for client authentication.";
           }
```

```
          container public-key {
            uses ks:private-key-grouping;
            description
              "A public key to be used for client authentication.";
          }
          leaf password {
            nacm:default-deny-all;
            type string;
            description
              "A password to be used for client authentication.";
          }
        }
      } // end client-auth

      container server-auth {
        must 'pinned-ssh-host-keys or pinned-ca-certs or '
             + 'pinned-server-certs';
        description
          "Trusted server identities.";
        leaf pinned-ssh-host-keys {
          type ks:pinned-host-keys;
          description
            "A reference to a list of SSH host keys used by the
             SSH client to authenticate SSH server host keys.
             A server host key is authenticated if it is an exact
             match to a configured SSH host key.";
        }

        leaf pinned-ca-certs {
          if-feature sshcom:ssh-x509-certs;
          type ks:pinned-certificates;
          description
            "A reference to a list of certificate authority (CA)
             certificates used by the SSH client to authenticate
             SSH server certificates.  A server certificate is
             authenticated if it has a valid chain of trust to
             a configured CA certificate.";
        }

        leaf pinned-server-certs {
          if-feature sshcom:ssh-x509-certs;
          type ks:pinned-certificates;
          description
            "A reference to a list of server certificates used by
             the SSH client to authenticate SSH server certificates.
             A server certificate is authenticated if it is an
             exact match to a configured server certificate.";
        }
```

```
      } // end server-auth

      container transport-params {
        if-feature ssh-client-transport-params-config;
        uses sshcom:transport-params-grouping;
        description
          "Configurable parameters for the SSH transport layer.";
      }

    }
  }
  <CODE ENDS>
```

4. The SSH Server Model

The SSH server model presented in this section contains one YANG grouping, for just the SSH-level configuration, omitting, for instance, configuration for which ports to open to listen for connections on.

This grouping references data nodes defined by the keystore model [I-D.ietf-netconf-keystore]. For instance, a reference to the keystore model is made to indicate which host key a server should present.

4.1. Tree Diagram

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

```
 module: ietf-ssh-server

   grouping ssh-server-grouping
     +---- server-identity
     |  +---- host-key* [name]
     |     +---- name?          string
     |     +---- (host-key-type)
     |        +--:(public-key)
     |        |  +---- public-key
     |        |     +---- algorithm?              identityref
     |        |     +---- private-key?            union
     |        |     +---- public-key?             binary
     |        |     +---x generate-private-key
     |        |        +---w input
     |        |           +---w algorithm    identityref
     |        +--:(certificate)
     |           +---- certificate {sshcom:ssh-x509-certs}?
     |              +---- algorithm?
     |              |       identityref
     |              +---- private-key?                           union
     |              +---- public-key?                            binary
     |              +---x generate-private-key
     |              |  +---w input
     |              |     +---w algorithm    identityref
     |              +---- 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
     +---- client-cert-auth {sshcom:ssh-x509-certs}?
     |  +---- pinned-ca-certs?       ks:pinned-certificates
     |  +---- pinned-client-certs?   ks:pinned-certificates
     +---- transport-params {ssh-server-transport-params-config}?
        +---- host-key
        |  +---- host-key-alg*   identityref
        +---- key-exchange
        |  +---- key-exchange-alg*   identityref
        +---- encryption
        |  +---- encryption-alg*   identityref
        +---- mac
        |  +---- mac-alg*   identityref
        +---- compression
           +---- compression-alg*   identityref
```

4.2. Example Usage

This section shows how it would appear if the ssh-server-grouping were populated with some data. This example is consistent with the examples presented in Section 2.2 of [I-D.ietf-netconf-keystore].

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

```
<!-- hypothetical example, as groupings don't have instance data -->\

<ssh-server xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server"
            xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common"\
>

  <!-- which host-keys will this SSH server present -->
  <server-identity>
    <host-key>
      <name>deployment-specific-certificate</name>
      <public-key>
        <algorithm xmlns:ks="urn:ietf:params:xml:ns:yang:ietf-keysto\
re">ks:secp521r1</algorithm>
        <private-key>base64encodedvalue==</private-key>
        <public-key>base64encodedvalue==</public-key>
      </public-key>
    </host-key>
  </server-identity>

  <!-- which client-certs will this SSH server trust -->
  <client-cert-auth>
    <pinned-ca-certs>deployment-specific-ca-certs</pinned-ca-certs>
    <pinned-client-certs>explicitly-trusted-client-certs</pinned-cli\
ent-certs>
  </client-cert-auth>

  <transport-params>
    <host-key>
      <host-key-alg>algs:ssh-rsa</host-key-alg>
    </host-key>
    <key-exchange>
      <key-exchange-alg>
        algs:diffie-hellman-group-exchange-sha256
      </key-exchange-alg>
    </key-exchange>
    <encryption>
      <encryption-alg>algs:aes256-ctr</encryption-alg>
      <encryption-alg>algs:aes192-ctr</encryption-alg>
      <encryption-alg>algs:aes128-ctr</encryption-alg>
      <encryption-alg>algs:aes256-cbc</encryption-alg>
```

```
        <encryption-alg>algs:aes192-cbc</encryption-alg>
        <encryption-alg>algs:aes128-cbc</encryption-alg>
      </encryption>
      <mac>
        <mac-alg>algs:hmac-sha2-256</mac-alg>
        <mac-alg>algs:hmac-sha2-512</mac-alg>
      </mac>
      <compression>
        <compression-alg>algs:none</compression-alg>
      </compression>
    </transport-params>

  </ssh-server>
```

4.3. YANG Module

This YANG module has a normative references to [RFC4253], [RFC6991], and [I-D.ietf-netconf-keystore].

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

  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-server";
  prefix "sshs";

  import ietf-ssh-common {
    prefix sshcom;
    revision-date 2017-10-30; // stable grouping definitions
    reference
      "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
  }

  import ietf-keystore {
    prefix ks;
    reference
      "RFC YYYY: Keystore Model";
  }

  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>
```

```
      Author:   Gary Wu
                <mailto:garywu@cisco.com>";


    description
     "This module defines a reusable grouping for a SSH server that
      can be used as a basis for specific SSH server instances.

      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 XXXX; see
      the RFC itself for full legal notices.";

    revision "2017-10-30" {
      description
       "Initial version";
      reference
       "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
    }

    // features

    feature ssh-server-transport-params-config {
      description
        "SSH transport layer parameters are configurable on an SSH
         server.";
    }

    // groupings

    grouping ssh-server-grouping {
      description
        "A reusable grouping for configuring a SSH server without
         any consideration for how underlying TCP sessions are
         established.";
      container server-identity {
        description
          "The list of host-keys the SSH server will present when
           establishing a SSH connection.";
        list host-key {
```

```
          key name;
          min-elements 1;
          ordered-by user;
          description
            "An ordered list of host keys the SSH server will use to
             construct its ordered list of algorithms, when sending
             its SSH_MSG_KEXINIT message, as defined in Section 7.1
             of RFC 4253.";
          reference
            "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
          leaf name {
            type string;
            description
              "An arbitrary name for this host-key";
          }
          choice host-key-type {
            mandatory true;
            description
              "The type of host key being specified";
            container public-key {
              uses ks:private-key-grouping;
              description
                "The SSH server uses a public-key for its host key.";
            }
            container certificate {
              if-feature sshcom:ssh-x509-certs;
              uses ks:private-key-grouping;
              uses ks:certificate-grouping;
              description
                "The SSH server uses a certificate for its host key.";
            }
          }
        }
      }

      container client-cert-auth {
        if-feature sshcom:ssh-x509-certs;
        description
          "A reference to a list of pinned certificate authority (CA)
           certificates and a reference to a list of pinned client
           certificates.";
        leaf pinned-ca-certs {
          type ks:pinned-certificates;
          description
            "A reference to a list of certificate authority (CA)
             certificates used by the SSH server to authenticate
             SSH client certificates.  A client certificate is
             authenticated if it has a valid chain of trust to
```

```
             a configured pinned CA certificate.";
        }
        leaf pinned-client-certs {
          type ks:pinned-certificates;
          description
            "A reference to a list of client certificates used by
             the SSH server to authenticate SSH client certificates.
             A clients certificate is authenticated if it is an
             exact match to a configured pinned client certificate.";
        }
      }

      container transport-params {
        if-feature ssh-server-transport-params-config;
        uses sshcom:transport-params-grouping;
        description
          "Configurable parameters for the SSH transport layer.";
      }

    }
  }
  <CODE ENDS>
```

5. The SSH Common Model

The SSH common model presented in this section contains identities and groupings common to both SSH clients and SSH servers. The transport-params-grouping can be used to configure the list of SSH transport algorithms permitted by the SSH client or SSH server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the SSH transport layer connection. The ability to restrict the the algorithms allowed is provided in this grouping for SSH clients and SSH servers that are capable of doing so and may serve to make SSH clients and SSH servers compliant with security policies.

Features are defined for algorithms that are OPTIONAL or are not widely supported by popular implementations. Note that the list of algorithms is not exhaustive. As well, some algorithms that are REQUIRED by [RFC4253] are missing, notably "ssh-dss" and "diffie-hellman-group1-sha1" due to their weak security and there being alternatives that are widely supported.

5.1. Tree Diagram

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

```
module: ietf-ssh-common

  grouping transport-params-grouping
    +---- host-key
    |  +---- host-key-alg*   identityref
    +---- key-exchange
    |  +---- key-exchange-alg*   identityref
    +---- encryption
    |  +---- encryption-alg*   identityref
    +---- mac
    |  +---- mac-alg*   identityref
    +---- compression
       +---- compression-alg*   identityref
```

5.2. Example Usage

This section shows how it would appear if the transport-params-grouping were populated with some data.

```
<!-- hypothetical example, as groupings don't have instance data -->
<transport-params
   xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-common"
   xmlns:algs="urn:ietf:params:xml:ns:yang:ietf-ssh-common">

  <host-key>
    <host-key-alg>algs:x509v3-rsa2048-sha256</host-key-alg>
    <host-key-alg>algs:ssh-rsa</host-key-alg>
  </host-key>
  <key-exchange>
    <key-exchange-alg>
      algs:diffie-hellman-group-exchange-sha256
    </key-exchange-alg>
  </key-exchange>
  <encryption>
    <encryption-alg>algs:aes256-ctr</encryption-alg>
    <encryption-alg>algs:aes192-ctr</encryption-alg>
    <encryption-alg>algs:aes128-ctr</encryption-alg>
    <encryption-alg>algs:aes256-cbc</encryption-alg>
    <encryption-alg>algs:aes192-cbc</encryption-alg>
    <encryption-alg>algs:aes128-cbc</encryption-alg>
  </encryption>
  <mac>
    <mac-alg>algs:hmac-sha2-256</mac-alg>
    <mac-alg>algs:hmac-sha2-512</mac-alg>
  </mac>
  <compression>
    <compression-alg>algs:none</compression-alg>
  </compression>

</transport-params>
```

5.3. YANG Module

This YANG module has a normative references to [RFC4344], [RFC4419], [RFC5656], and [RFC6668].

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

  namespace "urn:ietf:params:xml:ns:yang:ietf-ssh-common";
  prefix "sshcom";

  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>

      Author:   Gary Wu
                <mailto:garywu@cisco.com>";

    description
     "This module defines a common features, identities, and groupings
      for Secure Shell (SSH).

      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 XXXX; see
      the RFC itself for full legal notices.";

    revision "2017-10-30" {
      description
       "Initial version";
      reference
       "RFC XXXX: YANG Groupings for SSH Clients and SSH Servers";
    }

    // features

    feature ssh-ecc {
      description
        "Elliptic Curve Cryptography is supported for SSH.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    feature ssh-x509-certs {
      description
        "X.509v3 certificates are supported for SSH as per RFC 6187.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
```

```
                    Authentication";
    }

    feature ssh-dh-group-exchange {
      description
        "Diffie-Hellman Group Exchange is supported for SSH.";
      reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
                   Secure Shell (SSH) Transport Layer Protocol";
    }

    feature ssh-ctr {
      description
        "SDCTR encryption mode is supported for SSH.";
      reference
        "RFC 4344: The Secure Shell (SSH) Transport Layer
                   Encryption Modes";
    }

    feature ssh-sha2 {
      description
        "The SHA2 family of cryptographic hash functions is supported
         for SSH.";
      reference
        "FIPS PUB 180-4: Secure Hash Standard (SHS)";
    }

    feature ssh-zlib {
      description
        "ZLIB (LZ77) compression is supported for SSH.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    // identities

    identity public-key-alg-base {
      description
        "Base identity used to identify public key algorithms.";
    }

    identity ssh-dss {
      base public-key-alg-base;
      description
        "Digital Signature Algorithm using SHA-1 as the hashing
         algorithm.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
```

```
    }

    identity ssh-rsa {
      base public-key-alg-base;
      description
        "RSASSA-PKCS1-v1_5 signature scheme using SHA-1 as the hashing
         algorithm.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity ecdsa-sha2-nistp256 {
      base public-key-alg-base;
      if-feature "ssh-ecc and ssh-sha2";
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
         nistp256 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-nistp384 {
      base public-key-alg-base;
      if-feature "ssh-ecc and ssh-sha2";
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
         nistp384 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity ecdsa-sha2-nistp521 {
      base public-key-alg-base;
      if-feature "ssh-ecc and ssh-sha2";
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
         nistp521 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity x509v3-ssh-rsa {
      base public-key-alg-base;
      if-feature ssh-x509-certs;
      description
```

```
        "RSASSA-PKCS1-v1_5 signature scheme using a public key stored in
         an X.509v3 certificate and using SHA-1 as the hashing
         algorithm.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
                   Authentication";
    }

    identity x509v3-rsa2048-sha256 {
      base public-key-alg-base;
      if-feature "ssh-x509-certs and ssh-sha2";
      description
        "RSASSA-PKCS1-v1_5 signature scheme using a public key stored in
         an X.509v3 certificate and using SHA-256 as the hashing
         algorithm.  RSA keys conveyed using this format MUST have a
         modulus of at least 2048 bits.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
         Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp256 {
      base public-key-alg-base;
      if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
         nistp256 curve with a public key stored in an X.509v3
         certificate and using the SHA2 family of hashing algorithms.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
         Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp384 {
      base public-key-alg-base;
      if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
         nistp384 curve with a public key stored in an X.509v3
         certificate and using the SHA2 family of hashing algorithms.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
         Authentication";
    }

    identity x509v3-ecdsa-sha2-nistp521 {
      base public-key-alg-base;
      if-feature "ssh-ecc and ssh-x509-certs and ssh-sha2";
```

```
      description
        "Elliptic Curve Digital Signature Algorithm (ECDSA) using the
         nistp521 curve with a public key stored in an X.509v3
         certificate and using the SHA2 family of hashing algorithms.";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
         Authentication";
    }

    identity key-exchange-alg-base {
      description
        "Base identity used to identify key exchange algorithms.";
    }

    identity diffie-hellman-group14-sha1 {
      base key-exchange-alg-base;
      description
        "Diffie-Hellman key exchange with SHA-1 as HASH and
         Oakley Group 14 (2048-bit MODP Group).";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity diffie-hellman-group-exchange-sha1 {
      base key-exchange-alg-base;
      if-feature ssh-dh-group-exchange;
      description
        "Diffie-Hellman Group and Key Exchange with SHA-1 as HASH.";
      reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
                   Secure Shell (SSH) Transport Layer Protocol";
    }

    identity diffie-hellman-group-exchange-sha256 {
      base key-exchange-alg-base;
      if-feature "ssh-dh-group-exchange and ssh-sha2";
      description
        "Diffie-Hellman Group and Key Exchange with SHA-256 as HASH.";
      reference
        "RFC 4419: Diffie-Hellman Group Exchange for the
                   Secure Shell (SSH) Transport Layer Protocol";
    }

    identity ecdh-sha2-nistp256 {
      base key-exchange-alg-base;
      if-feature "ssh-ecc and ssh-sha2";
      description
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
```

```
         nistp256 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity ecdh-sha2-nistp384 {
      base key-exchange-alg-base;
      if-feature "ssh-ecc and ssh-sha2";
      description
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
         nistp384 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity ecdh-sha2-nistp521 {
      base key-exchange-alg-base;
      if-feature "ssh-ecc and ssh-sha2";
      description
        "Elliptic Curve Diffie-Hellman (ECDH) key exchange using the
         nistp521 curve and the SHA2 family of hashing algorithms.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
                   Secure Shell Transport Layer";
    }

    identity encryption-alg-base {
      description
        "Base identity used to identify encryption algorithms.";
    }

    identity triple-des-cbc {
      base encryption-alg-base;
      description
        "Three-key 3DES in CBC mode.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity aes128-cbc {
      base encryption-alg-base;
      description
       "AES in CBC mode, with a 128-bit key.";
      reference
       "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }
```

```
    identity aes192-cbc {
      base encryption-alg-base;
      description
        "AES in CBC mode, with a 192-bit key.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity aes256-cbc {
      base encryption-alg-base;
      description
        "AES in CBC mode, with a 256-bit key.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity aes128-ctr {
      base encryption-alg-base;
      if-feature ssh-ctr;
      description
        "AES in SDCTR mode, with 128-bit key.";
      reference
        "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
           Modes";
    }

    identity aes192-ctr {
      base encryption-alg-base;
      if-feature ssh-ctr;
      description
        "AES in SDCTR mode, with 192-bit key.";
      reference
        "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
           Modes";
    }

    identity aes256-ctr {
      base encryption-alg-base;
      if-feature ssh-ctr;
      description
        "AES in SDCTR mode, with 256-bit key.";
      reference
        "RFC 4344: The Secure Shell (SSH) Transport Layer Encryption
           Modes";
    }

    identity mac-alg-base {
      description
```

```
        "Base identity used to identify message authentication
         code (MAC) algorithms.";
    }

    identity hmac-sha1 {
      base mac-alg-base;
      description
        "HMAC-SHA1";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity hmac-sha2-256 {
      base mac-alg-base;
      if-feature "ssh-sha2";
      description
        "HMAC-SHA2-256";
      reference
        "RFC 6668: SHA-2 Data Integrity Verification for the
                   Secure Shell (SSH) Transport Layer Protocol";
    }

    identity hmac-sha2-512 {
      base mac-alg-base;
      if-feature "ssh-sha2";
      description
        "HMAC-SHA2-512";
      reference
        "RFC 6668: SHA-2 Data Integrity Verification for the
                   Secure Shell (SSH) Transport Layer Protocol";
    }

    identity compression-alg-base {
      description
        "Base identity used to identify compression algorithms.";
    }

    identity none {
      base compression-alg-base;
      description
        "No compression.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity zlib {
      base compression-alg-base;
      if-feature ssh-zlib;
```

```
      description
        "ZLIB (LZ77) compression.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    // groupings

    grouping transport-params-grouping {
      description
        "A reusable grouping for SSH transport parameters.
         For configurable parameters, a zero-element leaf-list of
         algorithms indicates the system default configuration for that
         parameter.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      container host-key {
        description
          "Parameters regarding host key.";
        leaf-list host-key-alg {
          type identityref {
            base public-key-alg-base;
          }
          ordered-by user;
          description
            "Host key algorithms in order of descending preference.";
        }
      }
      container key-exchange {
        description
          "Parameters regarding key exchange.";
        leaf-list key-exchange-alg {
          type identityref {
            base key-exchange-alg-base;
          }
          ordered-by user;
          description
            "Key exchange algorithms in order of descending
             preference.";
        }
      }
      container encryption {
        description
          "Parameters regarding encryption.";
        leaf-list encryption-alg {
          type identityref {
            base encryption-alg-base;
          }
```

```
          ordered-by user;
          description
            "Encryption algorithms in order of descending preference.";
        }
      }
      container mac {
        description
          "Parameters regarding message authentication code (MAC).";
        leaf-list mac-alg {
          type identityref {
            base mac-alg-base;
          }
          ordered-by user;
          description
            "MAC algorithms in order of descending preference.";
        }
      }
      container compression {
        description
          "Parameters regarding compression.";
        leaf-list compression-alg {
          type identityref {
            base compression-alg-base;
          }
          ordered-by user;
          description
            "Compression algorithms in order of descending preference.";
        }
      }
    }
  }
  <CODE ENDS>
```

6. Security Considerations

The YANG modules defined in this document are 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.

Since the modules defined in this document define only groupings, these considerations are primarily for the designers of other modules that use these groupings.

There are a number of data nodes defined in the YANG modules 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 all the modules defined in this draft are sensitive to write operations. For instance, the addition or removal of references to keys, certificates, trusted anchors, etc., can dramatically alter the implemented security policy. However, no NACM annotations are applied as the data SHOULD be editable by users other than a designated 'recovery session'.

Some of the readable data nodes in the YANG modules 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:

/client-auth/password: This node in the 'ietf-ssh-client' module is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. The only time this node should be returned 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 RPC 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:

NONE

7. IANA Considerations

7.1. The IETF XML Registry

This document registers three URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

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

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

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

7.2. The YANG Module Names Registry

This document registers three YANG modules in the YANG Module Names registry [RFC7950]. Following the format in [RFC7950], the the following registrations are requested:

```
name:         ietf-ssh-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-client
prefix:       sshc
reference:    RFC XXXX

name:         ietf-ssh-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-server
prefix:       sshs
reference:    RFC XXXX

name:         ietf-ssh-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-ssh-common
prefix:       sshcom
reference:    RFC XXXX
```

8. 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, Michal Vasko, and Bert Wijnen.

9. References

9.1. Normative References

[I-D.ietf-netconf-keystore]
Watsen, K., "Keystore Model", draft-ietf-netconf-keystore-02 (work in progress), June 2017.

[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>.

[RFC4344] Bellare, M., Kohno, T., and C. Namprempre, "The Secure Shell (SSH) Transport Layer Encryption Modes", RFC 4344, DOI 10.17487/RFC4344, January 2006, <https://www.rfc-editor.org/info/rfc4344>.

[RFC4419] Friedl, M., Provos, N., and W. Simpson, "Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol", RFC 4419, DOI 10.17487/RFC4419, March 2006, <https://www.rfc-editor.org/info/rfc4419>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <https://www.rfc-editor.org/info/rfc6187>.

[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>.

[RFC6668] Bider, D. and M. Baushke, "SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol", RFC 6668, DOI 10.17487/RFC6668, July 2012, <https://www.rfc-editor.org/info/rfc6668>.

[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>.

[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>.

9.2. 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.

[OPENSSH] "OpenSSH", 2016, <http://www.openssh.com>.

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

[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>.

[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>.

[RFC4254] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, DOI 10.17487/RFC4254, January 2006, <https://www.rfc-editor.org/info/rfc4254>.

[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>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[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>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

Appendix A. Change Log

A.1. 00 to 01

- o Noted that '0.0.0.0' and '::' might have special meanings.
- o Renamed "keychain" to "keystore".

A.2. 01 to 02

- o Removed the groupings 'listening-ssh-client-grouping' and 'listening-ssh-server-grouping'. Now modules only contain the transport-independent groupings.
- o Simplified the "client-auth" part in the ietf-ssh-client module. It now inlines what it used to point to keystore for.
- o Added cipher suites for various algorithms into new 'ietf-ssh-common' module.

A.3. 02 to 03

- o Removed 'RESTRICTED' enum from 'password' leaf type.
- o Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.
- o Fixed description statement for leaf 'trusted-ca-certs'.

A.4. 03 to 04

- o Change title to "YANG Groupings for SSH Clients and SSH Servers"
- o Added reference to RFC 6668
- o Added RFC 8174 to Requirements Language Section.
- o Enhanced description statement for ietf-ssh-server's "trusted-ca-certs" leaf.
- o Added mandatory true to ietf-ssh-client's "client-auth" 'choice' statement.
- o Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- o Updated YANG to use typedefs around leafrefs to common keystore paths

- Now inlines key and certificates (no longer a leafref to keystore)

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Gary Wu
Cisco Systems

EMail: garywu@cisco.com