NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2017

K. Watsen
Juniper Networks
G. Wu
Cisco Networks

November 3, 2016

SSH Client and Server Models
draft-ietf-netconf-ssh-client-server-01

Abstract

This document defines two YANG modules, one defines groupings for a generic SSH client and the other defines groupings for a generic SSH 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:

- draft-ietf-netconf-keystore

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

- "XXXX" --> the assigned RFC value for this draft
- "YYYY" --> the assigned RFC value for draft-ietf-netconf-keystore

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

- "2016-11-02" --> the publication date of this draft

The following two Appendix sections are to be removed prior to publication:

- Appendix A. Change Log

o Appendix B. Open Issues

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 http://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 7, 2017.

Copyright Notice

Copyright (c) 2016 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 (http://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. Terminology 3
1.2. Tree Diagrams 3
2. The SSH Client Model 4
2.1. Tree Diagram 4
2.2. Example Usage 6
2.3. YANG Model 6
3. The SSH Server Model 11
3.1. Tree Diagram 11
3.2. Example Usage 12
3.3. YANG Model 13
4. Security Considerations 17

5. IANA Considerations 17
5.1. The IETF XML Registry 17
5.2. The YANG Module Names Registry 17
6. Acknowledgements 18
7. References 18
7.1. Normative References 18
7.2. Informative References 18
Appendix A. Change Log 20
A.1. server-model-09 to 00 20
Appendix B. Open Issues 20
Authors' Addresses 20

1. Introduction

This document defines two YANG [RFC6020] modules, one defines groupings for a generic SSH client and the other defines groupings for a generic SSH 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 two YANG modules in this document each define two groupings. One grouping defines everything other than what's needed for the TCP [RFC793] protocol layer. The other grouping uses the first grouping while adding TCP layer specifics (e.g., addresses to connect to, ports to listen on, etc.). This separation is done in order to enable applications the opportunity to define their own strategy for how the underlying TCP connection is established. For instance, applications supporting NETCONF Call Home [draft-ietf-netconf-call-home] could use the first grouping for the SSH parts it provides, while adding data nodes for the reversed TCP layer.

1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Tree Diagrams

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

o Brackets "[" and "]" enclose list keys.

- Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.

- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).

- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. The SSH Client Model

The SSH client model presented in this section contains two YANG groupings, one for a client that initiates the underlying TCP connection and another for a client that has had the TCP connection opened for it already (e.g., call home).

Both of these groupings reference data nodes defined by the Keystore model [draft-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].

2.1. Tree Diagram

The following tree diagram presents the data model for the two groupings defined in the ietf-ssh-client module.

```
module: ietf-ssh-client
  groupings:
  initiating-ssh-client-grouping
      +---- server-auth
      |  +---- trusted-ssh-host-keys?   -> /ks:keystore/trusted-ssh-hos
t-keys/name
      |  +---- trusted-ca-certs?        -> /ks:keystore/trusted-certifi
cates/name {ssh-x509-certs}?
      |  +---- trusted-server-certs?    -> /ks:keystore/trusted-certifi
cates/name
      +---- client-auth
         +---- matches* [name]
            +---- name?                    string
            +---- match* [name]
            |  +---- name?                    string
            |  +---- trusted-ssh-host-keys?   -> /ks:keystore/trusted-s
sh-host-keys/name
            |  +---- trusted-ca-certs?        -> /ks:keystore/trusted-c
ertificates/name
            |  +---- trusted-server-certs?    -> /ks:keystore/trusted-c
ertificates/name
            +---- user-auth-credentials?   -> /ks:keystore/user-auth-cr
edentials/user-auth-credential/username

  listening-ssh-client-grouping
      +---- address?       inet:ip-address
      +---- port?          inet:port-number
      +---- server-auth
      |  +---- trusted-ssh-host-keys?   -> /ks:keystore/trusted-ssh-hos
t-keys/name
      |  +---- trusted-ca-certs?        -> /ks:keystore/trusted-certifi
cates/name {ssh-x509-certs}?
      |  +---- trusted-server-certs?    -> /ks:keystore/trusted-certifi
cates/name
      +---- client-auth
         +---- matches* [name]
            +---- name?                    string
            +---- match* [name]
            |  +---- name?                    string
            |  +---- trusted-ssh-host-keys?   -> /ks:keystore/trusted-s
sh-host-keys/name
            |  +---- trusted-ca-certs?        -> /ks:keystore/trusted-c
ertificates/name
            |  +---- trusted-server-certs?    -> /ks:keystore/trusted-c
ertificates/name
            +---- user-auth-credentials?   -> /ks:keystore/user-auth-cr
edentials/user-auth-credential/username
```

2.2. Example Usage

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

FIXME (how to do an example for a module that only has groupings?)

2.3. YANG Model

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

```
<CODE BEGINS> file "ietf-ssh-client@2016-11-02.yang"

module ietf-ssh-client {
  yang-version 1.1;

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

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

  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>

    WG Chair: Mehmet Ersue
              <mailto:mehmet.ersue@nsn.com>

    WG Chair: Mahesh Jethanandani
              <mailto:mjethanandani@gmail.com>
```

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) 2014 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 "2016-11-02" {
    description
     "Initial version";
    reference
     "RFC XXXX: SSH Client and Server Models";
  }

  feature ssh-x509-certs {
    description
      "The ssh-x509-certs feature indicates that the SSH
       client supports RFC 6187";
    reference
      "RFC 6187: X.509v3 Certificates for Secure Shell
       Authentication";
  }

  grouping initiating-ssh-client-grouping {
    description
      "A reusable grouping for a SSH client that initiates the
       underlying TCP transport connection.";

    container server-auth {
      description
        "Trusted server identities.";
```

```
        leaf trusted-ssh-host-keys {
          type leafref {
            path "/ks:keystore/ks:trusted-ssh-host-keys/ks:name";
          }
          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 authenticate if it is an exact
             match to a configured trusted SSH host key.";
        }

        leaf trusted-ca-certs {
          if-feature ssh-x509-certs;
          type leafref {
            path "/ks:keystore/ks:trusted-certificates/ks:name";
          }
          description
            "A reference to a list of certificate authority (CA)
             certificates used by the SSH client to authenticate
             SSH server certificates.";
        }

        leaf trusted-server-certs {
          type leafref {
            path "/ks:keystore/ks:trusted-certificates/ks:name";
          }
          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 trusted server certificate.";
        }
      }

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

        list matches {
          key name;
          description
            "A matches expression, which performs like a firewall
             rulebase in that each matches expression is considered
             for a match before moving onto the next matches
             expression.  The first matching expression terminates
             the search, and its 'user-auth-credentials' are used
             to log into the SSH server.";
```

```
          leaf name {
            type string;
            description
              "An arbitrary name for this matches expression.";
          }
          list match {
            key name;
            description
              "A match rule.  The presented SSH server's host key
               is matched against possible trusted SSH host keys
               and certificates.  If a match is found, the specified
               'user-auth-credentials' is used to log into the
               SSH server.";
            leaf name {
              type string;
              description
                "An arbitrary name for this match rule.";
            }
            leaf trusted-ssh-host-keys {
              type leafref {
                path "/ks:keystore/ks:trusted-ssh-host-keys/ks:name";
              }
              description
                "A test to see if the presented SSH host key
                 matches any of the host keys in the specified
                 'trusted-ssh-host-keys' list maintained by the
                 keystore module.";
            }
            leaf trusted-ca-certs {
              type leafref {
                path "/ks:keystore/ks:trusted-certificates/ks:name";
              }
              description
                "A test to see if the presented SSH host key matches
                 any of the trusted CA certificates in the specified
                 'trusted-certificates' list maintained by the
                 keystore module.";
            }
            leaf trusted-server-certs {
              type leafref {
                path "/ks:keystore/ks:trusted-certificates/ks:name";
              }
              description
                "A test to see if the presented SSH host key matches
                 any of the trusted server certificates in the specified
                 'trusted-certificates' list maintained by the
                 keystore module.";
            }
```

```
          }
          leaf user-auth-credentials {
            type leafref {
              path "/ks:keystore/ks:user-auth-credentials/"
                   + "ks:user-auth-credential/ks:username";
            }
            description
              "The specific user authentication credentials to use if
               all of the above 'match' expressions match.";
          }
        }
      }
    }  // end initiating-ssh-client-grouping


    grouping listening-ssh-client-grouping {
      description
        "A reusable grouping for a SSH client that does not
         the underlying TCP transport connection have been
         established using some other mechanism.";
      leaf address {
        type inet:ip-address;
        description
         "The IP address to listen for call-home connections on.";
      }
      leaf port {
        type inet:port-number;
        description
         "The port number to listen for call-home connections.
          When this grouping is used, it is RECOMMENED that
          refine statement is used to either set a default port
          value or to set mandatory true.";
      }
      uses initiating-ssh-client-grouping;
    }



  }



  <CODE ENDS>
```

3. The SSH Server Model

The SSH server model presented in this section contains two YANG groupings, one for a server that opens a socket to accept TCP connections and another for a server that has had the TCP connection opened for it already (e.g., inetd).

Both of these groupings reference data nodes defined by the Keystore model [draft-ietf-netconf-keystore]. For instance, a reference to the keystore model is made to indicate which host key a server should present.

3.1. Tree Diagram

The following tree diagram presents the data model for the two groupings defined in the ietf-ssh-server module.

```
module: ietf-ssh-server
  groupings:
  listening-ssh-server-grouping
      +---- address?            inet:ip-address
      +---- port?               inet:port-number
      +---- host-keys
      |  +---- host-key* [name]
      |     +---- name?          string
      |     +---- (host-key-type)
      |        +--:(public-key)
      |        |  +---- public-key?    -> /ks:keystore/private-keys/pri
vate-key/name
      |        +--:(certificate)
      |           +---- certificate?   -> /ks:keystore/private-keys/pri
vate-key/certificate-chains/certificate-chain/name {ssh-x509-certs}?
      +---- client-cert-auth {ssh-x509-certs}?
         +---- trusted-ca-certs?       -> /ks:keystore/trusted-certific
ates/name
         +---- trusted-client-certs?   -> /ks:keystore/trusted-certific
ates/name

  non-listening-ssh-server-grouping
      +---- host-keys
      |  +---- host-key* [name]
      |     +---- name?          string
      |     +---- (host-key-type)
      |        +--:(public-key)
      |        |  +---- public-key?    -> /ks:keystore/private-keys/pri
vate-key/name
      |        +--:(certificate)
      |           +---- certificate?   -> /ks:keystore/private-keys/pri
vate-key/certificate-chains/certificate-chain/name {ssh-x509-certs}?
      +---- client-cert-auth {ssh-x509-certs}?
         +---- trusted-ca-certs?       -> /ks:keystore/trusted-certific
ates/name
         +---- trusted-client-certs?   -> /ks:keystore/trusted-certific
ates/name
```

3.2. Example Usage

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

```
   <listening-ssh-server
     xmlns="urn:ietf:params:xml:ns:yang:ietf-ssh-server">
     <port>830</port>
     <host-keys>
       <host-key>
         <name>deployment-specific-certificate</name>
         <certificate>ex-key-sect571r1-cert</certificate>
       </host-key>
     </host-keys>
     </certificates>
     <client-cert-auth>
       <trusted-ca-certs>
         deployment-specific-ca-certs
       </trusted-ca-certs>
       <trusted-client-certs>
         explicitly-trusted-client-certs
       </trusted-client-certs>
     </client-cert-auth>
   </listening-ssh-server>
```

3.3. YANG Model

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

```
  <CODE BEGINS> file "ietf-ssh-server@2016-11-02.yang"

  module ietf-ssh-server {
    yang-version 1.1;

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

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

    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>

      WG Chair: Mehmet Ersue
                <mailto:mehmet.ersue@nsn.com>

      WG Chair: Mahesh Jethanandani
                <mailto:mjethanandani@gmail.com>

      Editor:   Kent Watsen
                <mailto:kwatsen@juniper.net>";


    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) 2014 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 "2016-11-02" {
      description
       "Initial version";
      reference
       "RFC XXXX: SSH Client and Server Models";
    }

    // features
    feature ssh-x509-certs {
      description
        "The ssh-x509-certs feature indicates that the NETCONF
         server supports RFC 6187";
      reference
        "RFC 6187: X.509v3 Certificates for Secure Shell
         Authentication";
    }
```

```
    // grouping
    grouping non-listening-ssh-server-grouping {
      description
        "A reusable grouping for a SSH server that can be used as a
         basis for specific SSH server instances.";

      container host-keys {
        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";
            leaf public-key {
              type leafref {
                path "/ks:keystore/ks:private-keys/ks:private-key/"
                     + "ks:name";
              }
              description
                "The public key is actually identified by the name of
                 its cooresponding private-key in the keystore.";
            }
            leaf certificate {
              if-feature ssh-x509-certs;
              type leafref {
                path "/ks:keystore/ks:private-keys/ks:private-key/"
                     + "ks:certificate-chains/ks:certificate-chain/"
                     + "ks:name";
              }
              description
                "The name of a certificate in the keystore.";
```

```
            }
          }
        }
      }

      container client-cert-auth {
        if-feature ssh-x509-certs;
        description
          "A reference to a list of trusted certificate authority (CA)
           certificates and a reference to a list of trusted client
           certificates.";
        leaf trusted-ca-certs {
          type leafref {
            path "/ks:keystore/ks:trusted-certificates/ks:name";
          }
          description
            "A reference to a list of certificate authority (CA)
             certificates used by the SSH server to authenticate
             SSH client certificates.";
        }

        leaf trusted-client-certs {
          type leafref {
            path "/ks:keystore/ks:trusted-certificates/ks:name";
          }
          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 trusted client certificate.";
        }
      }
    }


    grouping listening-ssh-server-grouping {
      description
        "A reusable grouping for a SSH server that can be used as a
         basis for specific SSH server instances.";
      leaf address {
        type inet:ip-address;
        description
         "The IP address of the interface to listen on.  The SSH
          server will listen on all interfaces if no value is
          specified.  Please note that some addresses have special
          meanings (e.g., '0.0.0.0' and '::').";
      }
      leaf port {
```

```
        type inet:port-number;
        description
         "The local port number on this interface the SSH server
          listens on.  When this grouping is used, it is RECOMMENED
          that refine statement is used to either set a default port
          value or to set mandatory true.";
      }
      uses non-listening-ssh-server-grouping;
    }
  }
```

<CODE ENDS>

4. Security Considerations

5. IANA Considerations

5.1. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC2119]. 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.
```

5.2. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], 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
```

6. 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, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, Juergen Schoenwaelder, Phil Shafer, Sean Turner, Michal Vasko, and Bert Wijnen.

7. References

7.1. Normative References

[draft-ietf-netconf-keystore]
Watsen, K., "Keystore Model", draft-ieft-netconf-keystore-00 (work in progress), 2016, <https://datatracker.ietf.org/html/draft-ieft-netconf-keystore>.

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

7.2. Informative References

[draft-ietf-netconf-call-home]
Watsen, K., "NETCONF Call Home and RESTCONF Call Home", draft-ieft-netconf-call-home-17 (work in progress), 2015, <https://datatracker.ietf.org/html/draft-ieft-netconf-call-home-17>.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc4254>.

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

[RFC793] Postel, J., "TRANSMISSION CONTROL PROTOCOL", STD 7, September 1981, <https://www.ietf.org/rfc/rfc793.txt>.

Appendix A. Change Log

A.1. server-model-09 to 00

- This draft was split out from draft-ietf-netconf-server-model-09.
- Added in previously missing ietf-ssh-client module.
- Noted that '0.0.0.0' and '::' might have special meanings.

Appendix B. Open Issues

Please see: https://github.com/netconf-wg/ssh-client-server/issues.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Gary Wu
Cisco Networks

EMail: garywu@cisco.com