

NETCONF Working Group
Internet-Draft
Obsoletes: 5539 (if approved)
Intended status: Standards Track
Expires: October 30, 2012

M. Badra
Dhofar University
April 28, 2012

NETCONF Over Transport Layer Security (TLS)
draft-badra-netconf-rfc5539bis-02

Abstract

The Network Configuration Protocol (NETCONF) provides mechanisms to install, manipulate, and delete the configuration of network devices. This document describes how to use the Transport Layer Security (TLS) protocol to secure NETCONF exchanges. This document obsoletes RFC 5539.

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 October 30, 2012.

Copyright Notice

Copyright (c) 2012 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. Conventions Used in This Document	3
2. NETCONF over TLS	3
2.1. Connection Initiation	3
2.2. Connection Closure	4
3. Endpoint Authentication, Identification and Authorization . .	4
3.1. Server Identity	4
3.2. Client Identity	5
3.2.1. Deriving NETCONF Usernames From NETCONF Client Certificates	5
4. Security Considerations	16
5. IANA Considerations	16
6. Acknowledgements	17
7. Contributor's Address	17
8. References	18
8.1. Normative References	18
8.2. Informative References	18
Appendix A. Change Log (to be removed by RFC Editor before publication)	19
A.1. From -01 to -02	19
A.2. From -00 to -01	19
A.3. From RFC5539 to draft-badra-netconf-rfc5539bis-00	19
Author's Address	19

1. Introduction

The NETCONF protocol [RFC6241] defines a mechanism through which a network device can be managed. NETCONF is connection-oriented, requiring a persistent connection between peers. This connection must provide integrity, confidentiality, peer authentication, and reliable, sequenced data delivery.

This document defines "NETCONF over TLS", which includes support for certificate and pre-shared key (PSK)-based authentication and key derivation, utilizing the protected ciphersuite negotiation, mutual authentication, and key management capabilities of the TLS (Transport Layer Security) protocol, described in [RFC5246].

1.1. Conventions Used in This Document

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

2. NETCONF over TLS

Since TLS is application-protocol-independent, NETCONF can operate on top of the TLS protocol transparently. This document defines how NETCONF can be used within a TLS session.

2.1. Connection Initiation

The peer acting as the NETCONF client MUST also act as the TLS client. The client actively opens the TLS connection and the server passively listens for the incoming TLS connection on the TCP port 6513. It MUST therefore send the TLS ClientHello message to begin the TLS handshake. Once the TLS handshake has finished, the client and the server MAY begin to exchange NETCONF data. In particular, the client will send complete XML documents to the server containing <rpc> elements, and the server will respond with complete XML documents containing <rpc-reply> elements. The client MAY indicate interest in receiving event notifications from a server by creating a subscription to receive event notifications [RFC5277]. In this case, the server replies to indicate whether the subscription request was successful and, if it was successful, the server begins sending the event notifications to the client as the events occur within the system.

All NETCONF messages MUST be sent as TLS "application data". It is possible that multiple NETCONF messages be contained in one TLS record, or that a NETCONF message be transferred in multiple TLS

records.

The previous version [RFC5539] of this document used the same framing sequence defined in [RFC6242], under the assumption that it could not be found in well-formed XML documents. However, this assumption is not correct [RFC6242]. In order to solve this problem, and at the same time be compatible with existing implementations, this document uses the framing protocol defined in [RFC6242] as following :

The <hello> message MUST be followed by the character sequence]]>]]. Upon reception of the <hello> message, the receiving peer's TLS Transport layer conceptually passes the <hello> message to the Messages layer. If the :base:1.1 capability is advertised by both peers, the chunked framing mechanism defined in Section 4.2 of [RFC6242] is used for the remainder of the NETCONF session. Otherwise, the old end-of-message-based mechanism (see Section 4.3 of [RFC6242]) is used.

Implementation of the protocol specified in this document MAY implement any TLS cipher suite that provides mutual authentication [RFC5246].

Implementations MUST support TLS 1.2 [RFC5246] and are REQUIRED to support the mandatory-to-implement cipher suite, which is TLS_RSA_WITH_AES_128_CBC_SHA. This document is assumed to apply to future versions of TLS; in which case, the mandatory-to-implement cipher suite for the implemented version MUST be supported.

2.2. Connection Closure

Exiting NETCONF is accomplished using the <close-session> operation. A NETCONF server will process NETCONF messages from the NETCONF client in the order in which they are received. When the NETCONF server processes a <close-session> operation, the NETCONF server SHALL respond and close the TLS session channel. The NETCONF server MUST NOT process any NETCONF messages received after the <close-session> operation. The TLS session is closed as described in [RFC6242] Section 7.2.1.

3. Endpoint Authentication, Identification and Authorization

3.1. Server Identity

If the server's presented certificate has passed certification path validation [RFC5280] to a configured trust anchor, the client MUST carefully examine the certificate presented by the server to determine if it meets the client's expectations. Particularly, the

client MUST check its understanding of the server hostname against the server's identity as presented in the server Certificate message, in order to prevent man-in-the-middle attacks.

Matching is performed according to the rules and guidelines defined in [RFC6125].

If the match fails, the client MUST either ask for explicit user confirmation or terminate the connection and indicate the server's identity is suspect.

Additionally, clients MUST verify the binding between the identity of the servers to which they connect and the public keys presented by those servers. Clients SHOULD implement the algorithm in Section 6 of [RFC5280] for general certificate validation, but MAY supplement that algorithm with other validation methods that achieve equivalent levels of verification (such as comparing the server certificate against a local store of already-verified certificates and identity bindings).

If the client has external information as to the expected identity of the server, the hostname check MAY be omitted.

3.2. Client Identity

The server MUST verify the identity of the client to ensure that the incoming client request is legitimate before any configuration or state data is sent to or received from the client.

The NETCONF protocol [RFC6241] requires that the transport protocol's authentication process MUST result in an authenticated client identity whose permissions are known to the server. The authenticated identity of a client is commonly referred to as the NETCONF username.

The username provided by the TLS implementation will be made available to the NETCONF message layer as the NETCONF username without modification. If the username does not comply to the NETCONF requirements on usernames [RFC6241], i.e., the username is not representable in XML, the TLS session MUST be dropped.

Algorithms for mapping certificates or PSK identities (sent by the client) to NETCONF usernames are described below.

3.2.1. Deriving NETCONF Usernames From NETCONF Client Certificates

The algorithm for deriving NETCONF usernames from TLS certificates is patterned after the algorithm for deriving `tmSecurityNames` from TLS

certificates specified in Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP) [RFC6353]. The NETCONF server MUST implement the algorithms for deriving NETCONF usernames from presented certificates that are documented in the ietf-netconf-tls YANG module. This YANG module lets the NETCONF security administrator configure how the NETCONF server derives NETCONF usernames from presented certificates. It also lets different certificate-to-username derivation algorithms be used for different certificates.

When a NETCONF server accepts a TLS connection from a NETCONF client, the NETCONF server attempts to derive a NETCONF username from the certificate presented by the NETCONF client. If the NETCONF server cannot derive a valid NETCONF username from the client's presented certificate, then the NETCONF server MUST close the TLS connection, and MUST NOT accept NETCONF messages over it. The NETCONF server MAY use any of the following algorithms to produce the NETCONF username from the certificate presented by the NETCONF client:

- o Map a certificate directly to a specified, pre-configured, NETCONF username;
- o Extract the subjectAltName's rfc822Name from the certificate, then use the extracted rfc822Name as the NETCONF username;
- o Extract the subjectAltName's dnsName from the certificate, then use the extracted dnsName as the NETCONF username;
- o Extract the subjectAltName's ipAddress from the certificate, then use the extracted ipAddress as the NETCONF username;
- o Examine the subjectAltName's rfc822Name, dnsName, and ipAddress fields in a pre-defined order. Return the value from the first subjectAltName field that is examined, defined, and populated with a non-empty value. If no subjectAltName field of a specific type is defined, then the examination skips that field and proceeds to examine the next field type. If a subjectAltName field is defined, but the value is not populated, or is populated by an empty value, then the examination skips that field and proceeds to examine the next field type.

The NETCONF server MUST implement all of these algorithms, and allow the deployer to choose the algorithm used. The certificate-to-username-transforms container in the ietf-netconf-tls YANG module specifies how a NETCONF server transforms a certificate into a NETCONF username.

3.2.1.1. Identifying a Certificate

A client certificate has an identity: the certificate. The TLS and corresponding protocols provide an identity. The identity shows that "this client certificate has shown that it, indeed, is on the other side of the connection". With a complete certificate, the certificate receiver can be certain that for someone or something on the other side to use that certificate successfully, it has the associated private key.

The problem with using the entire certificates as the identity is that they are difficult for people to use. It is generally accepted that a fingerprint of a certificate is not likely to come up with a collision against a fingerprint of another (different) certificate. Thus, assuming a good hash algorithm, a fingerprint can be a safe short-hand for identifying a certificate.

If a locally held copy of a trusted CA certificate is configured in the transformation container, and that CA certificate was used to validate the path to the presented certificate, then the NETCONF server SHOULD use that list entry in the transformation container. All presented certificates validated by the configured CA certificate will be transformed to NETCONF usernames using the same transformation algorithm.

3.2.1.2. Deriving NETCONF Usernames From PSK identities

Implementations MAY optionally support TLS Pre-Shared Key (PSK) authentication [RFC4279]. RFC4279 describes pre-shared key ciphersuites for TLS. During the TLS Handshake, the client indicates which key to use by including a "PSK identity" in the TLS ClientKeyExchange message [RFC4279]. On the server side, this PSK identity is used to look up the key corresponding to the presented PSK identity. If the selected pre-shared keys match and the key is valid, then the client is authenticated and the NETCONF username associated with the PSK identity. For details on how the PSK identity MAY be encoded in UTF-8, see section 5.1. of RFC [RFC6241].

3.2.1.3. Remote Configuration

The ietf-netconf-tls YANG module defines objects for remotely configuring the mapping of TLS certificates and of PSK Identities to NETCONF usernames.

```
module ietf-netconf-tls {
```

```
namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-tls";

prefix "nctls";

import ietf-yang-types {
  prefix yang;
}

import ietf-netconf-acm {
  prefix nacm;
}

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: Bert Wijnen
            <mailto:bertietf@bwijnen.net>

  Editor:    Mohamad Badra
            <mailto:mbadra@gmail.com>";

description
  "This module applies to NETCONF over TLS.  It specifies how NETCONF
  servers transform X.509 certificates presented by clients into
  NETCONF usernames.  It also specifies how NETCONF servers transform
  pre-shared TLS keys into NETCONF usernames.

  This YANG module is patterned after parts of the SNMP-TLS-TM-MIB
  defined in RFC 6353.  Much of the description text has been copied
  directly from the SNMP-TLS-TM-MIB, and modified as necessary.

  Copyright (c) 2012 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.";
// RFC Ed.: replace XXXX with actual RFC number and
// remove this note

// RFC Ed.: please update the date to the date of publication

revision "2012-04-11" {
  description
    "Incorporates comments on draft-badra-netconf-rfc5539bis-01.txt.";
  reference
    "RFC XXXX: NETCONF over Transport Layer Security (TLS)";
}

revision "2012-02-13" {
  description
    "Initial version";
  reference
    "RFC XXXX: NETCONF over Transport Layer Security (TLS)";
}

feature map-certificates {
  description
    "The :map-certificates capability implements mapping X.509
    certificates to NETCONF user names.";
}

feature map-pre-shared-keys {
  description
    "The :map-pre-shared-keys capability implements mapping TLS
    pre-shared keys to NETCONF user names.";
}

typedef tls-fingerprint-type {
  type string {
    pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2})*';
  }
  description
    "A cryptographic signature (fingerprint) value that can be used to
    uniquely reference other data of potentially arbitrary length.";
}

//
//  Objects related to deriving NETCONF usernames from X.509
```

```
// certificates.  
//
```

```
container cert-mappings {  
  if-feature map-certificates;  
  config true;  
  description
```

```
    "This container is used by a NETCONF server to map the NETCONF  
    client's presented X.509 certificate to a NETCONF username.
```

On an incoming TLS connection, the client's presented certificate MUST either be validated based on an established trust anchor, or it MUST directly match a fingerprint in this container. This container does not provide any mechanisms for configuring the trust anchors; the transfer of any needed trusted certificates for certificate chain validation is expected to occur through an out-of-band transfer.

Once the certificate has been found acceptable (either by certificate chain validation or directly matching a fingerprint in this container), this container is consulted to determine the appropriate NETCONF username to associate with the remote connection. This is done by considering each list entry from this container in prioritized order according to its index value. Each list entry's fingerprint value determines whether the list entry is a match for the incoming connection:

- 1) If the list entry's fingerprint value matches that of the presented certificate, then consider the list entry as a successful match.
- 2) If the list entry's fingerprint value matches that of a locally held copy of a trusted CA certificate, and that CA certificate was part of the CA certificate chain to the presented certificate, then consider the list entry as a successful match.

This feature lets the NETCONF server derive NETCONF usernames from all certificates signed by the trusted CA certificate. The NETCONF server will derive all NETCONF usernames using the same derivation algorithm. The NETCONF server requires only a single list entry to configure this behavior.

Once a matching list entry has been found, the NETCONF server uses the map-type value to determine how the NETCONF username associated with the session should be determined. See the map-type leaf's description for details on determining the NETCONF

username value. If it is impossible to determine a NETCONF username from the list entry's data combined with the data presented in the certificate, then additional list entries MUST be searched looking for another potential match. If a resulting NETCONF username mapped from a given list entry is not compatible with the needed requirements of a NETCONF username, then it MUST be considered an invalid match and additional list entries MUST be searched looking for another potential match.

If no matching and valid list entry can be found, then the NETCONF server MUST close the connection, and MUST NOT accept NETCONF messages over it.

Non-consecutive values of index are acceptable and implementations should continue to the next highest numbered list entry. It is recommended that administrators skip index values to leave room for the insertion of future list entries (for example, use values of 10 and 20 when creating initial list entries).

Security administrators are encouraged to make use of certificates with subjectAltName fields that can be used as NETCONF usernames so that a single root CA certificate can allow all child certificate's subjectAltName to map directly to a NETCONF usernames via a 1:1 transformation. However, this container is flexible to allow for situations where existing deployed certificate infrastructures do not provide adequate subjectAltName values for use as NETCONF usernames.";

```
list cert-map {
  key "index";
  description
    "A single list entry that specifies a mapping for an incoming
    TLS certificate to a NETCONF username.";

  leaf index {
    type uint32 {
      range "1..4294967295";
    }
    description
      "A unique, prioritized index for the given entry. Lower
      numbers indicate a higher priority.";
  }

  container fingerprint {
    choice algorithm-and-hash {
      leaf md5 {
        type tls-fingerprint-type;
      }
    }
  }
}
```

```
    }
    leaf sha1 {
      type tls-fingerprint-type;
    }
    leaf sha224 {
      type tls-fingerprint-type;
    }
    leaf sha256 {
      type tls-fingerprint-type;
    }
    leaf sha384 {
      type tls-fingerprint-type;
    }
    leaf sha512 {
      type tls-fingerprint-type;
    }
  }
  description
    "Specifies the signature algorithm and cryptographic
    signature (fingerprint) used to identify an X.509
    certificate.

    Implementations of this YANG module MAY, but are not
    required to, implement all of these cryptographic signature
    algorithms. Implementations of this YANG module MUST
    implement at least one of these cryptographic signature
    algorithms.

    The available choices may be extended in the future as
    stronger cryptographic signature algorithms become
    available and are deemed necessary."

  reference
    "RFC 5246: The Transport Layer Security (TLS) Protocol
    Version 1.2; Section 7.4.1.4.1, Signature Algorithms";
} // choice algorithm-and-hash
} // container fingerprint

leaf map-type {
  type enumeration {
    enum specified { value 1; }
    enum rfc822Name { value 2; }
    enum dnsName { value 3; }
    enum ipAddress { value 4; }
    enum rfc822Name-dnsName-ipAddress { value 5; }
  }
}

description
```

"Specifies the algorithm for deriving a NETCONF username from a certificate. If a mapping succeeds, then it will return a NETCONF username.

If the resulting mapped value is not compatible with the needed requirements of a NETCONF username, then subsequent list entries MUST be searched for additional NETCONF username matches to look for a mapping that succeeds.

For each enumerated value listed above, the NETCONF server derives the NETCONF from the presented client certificate as described:

specified

Directly specifies the NETCONF username to be used for this certificate. The value of the NETCONF username to use is specified in the data leaf of the list. The data leaf MUST contain a non-zero length value or the mapping described in this list entry MUST be considered a failure.

rfc822Name

Maps a subjectAltName's rfc822Name to a NETCONF username. The local part of the rfc822Name is passed unaltered but the host-part of the name MUST be passed in lowercase. This mapping results in a 1:1 correspondence between equivalent subjectAltName rfc822Name values and NETCONF username values except that the host-part of the name MUST be passed in lowercase.

Example rfc822Name Field: FooBar@Example.COM
is mapped to NETCONF username: FooBar@example.com.

dnsName

Maps a subjectAltName's dnsName to a NETCONF username after first converting it to all lowercase (RFC 5280 does not specify converting to lowercase so this involves an extra step). This mapping results in a 1:1 correspondence between subjectAltName dnsName values and the NETCONF username values.

reference: RFC 5280 - Internet X.509 Public Key
Infrastructure Certificate and Certificate
Revocation List (CRL) Profile.

`ipAddress`

Maps a `subjectAltName`'s `ipAddress` to a NETCONF username by transforming the binary encoded address as follows:

- 1) for IPv4, the value is converted into a decimal-dotted quad address (e.g., '192.0.2.1').
- 2) for IPv6 addresses, the value is converted into a 32-character all lowercase hexadecimal string without any colon separators.

This mapping results in a 1:1 correspondence between `subjectAltName ipAddress` values and the NETCONF username values.

`rfc822Name-dnsName-ipAddress`

The NETCONF server derives the NETCONF username from the `subjectAltName` fields `rfc822Name`, `dnsName`, and `ipAddress`, as described in sections above. This enumeration specifies the NETCONF server first examines the `rfc822Name`, then examines the `dnsName`, then finally examines the `ipAddress`. The first matching `subjectAltName` value found in the certificate MUST be used when deriving the NETCONF username.

These mappings result in a 1:1 correspondence between `subjectAltName` values and NETCONF username values. The sub-mapping algorithms produced by these combined algorithms cannot produce conflicting results between themselves.";

```

} // leaf map-type

leaf data {
  type string {
    length "1..max";
  }
  description
    "Auxiliary data used as optional configuration information for
    a given mapping specified by the map-type leaf. Only some
    mapping systems will make use of this leaf. When the NETCONF
    server derives the NETCONF username from the client's presented
    certificate, the value in this leaf MUST be ignored for any
    mapping type that does not require data present in this leaf.";
}
} // list cert-map
} // container cert-mappings

```

```
//
//  Objects related to deriving NETCONF usernames from TLS pre-shared
//  keys.
//

container psk-map {
  if-feature map-pre-shared-keys;
  list psk {
    key psk-identity;

    leaf psk-identity {
      type string;
      description
        "The PSK identity encoded as a UTF-8 string.";
      reference
        "RFC 4279: Pre-Shared Key Ciphersuites for Transport Layer
          Security (TLS)";
    }

    leaf user-name {
      type nacm:user-name-type;
      mandatory true;
      description
        "The NETCONF username associated with this PSK identity.";
    }

    leaf valid-not-before {
      type yang:date-and-time;
      description
        "This PSK identity is not valid before the given data
          and time.";
    }

    leaf valid-not-after {
      type yang:date-and-time;
      description
        "This PSK identity is not valid before the given data
          and time.";
    }

    leaf key {
      type string;
      pattern '([0-9a-fA-F]){2}(:([0-9a-fA-F]){2})*';
      nacm:default-deny-all;
      description
        "The key associated with the PSK identity";
    }
  }
}
```

```
}    // container psk-map  
}
```

4. Security Considerations

The security considerations described throughout [RFC5246] and [RFC6241] apply here as well.

This document in its current version does not support third-party authentication (e.g., backend Authentication, Authorization, and Accounting (AAA) servers) due to the fact that TLS does not specify this way of authentication and that NETCONF depends on the transport protocol for the authentication service. If third-party authentication is needed, BEEP or SSH transport can be used.

An attacker might be able to inject arbitrary NETCONF messages via some application that does not carefully check exchanged messages. When the :base:1.1 capability is not advertised by both peers, an attacker might be able to deliberately insert the delimiter sequence]]>]] in a NETCONF message to create a DoS attack. If the :base:1.1 capability is not advertised by both peers, applications and NETCONF APIs MUST ensure that the delimiter sequence]]>]] never appears in NETCONF messages; otherwise, those messages can be dropped, garbled, or misinterpreted. More specifically, if the delimiter sequence is found in a NETCONF message by the sender side, a robust implementation of this document SHOULD warn the user that illegal characters have been discovered. If the delimiter sequence is found in a NETCONF message by the receiver side (including any XML attribute values, XML comments, or processing instructions), a robust implementation of this document MUST silently discard the message without further processing and then stop the NETCONF session.

Finally, this document does not introduce any new security considerations compared to [RFC6242].

5. IANA Considerations

Based on the previous version of this document, RFC 5539, IANA has assigned a TCP port number (6513) in the "Registered Port Numbers" range with the name "netconf-tls". This port will be the default port for NETCONF over TLS, as defined in this document.

Registration Contact: Mohamad Badra, mbadra@gmail.com.
Transport Protocol: TCP.
Port Number: 6513
Broadcast, Multicast or Anycast: No.
Port Name: netconf-tls.
Service Name: netconf.
Reference: RFC 5539

6. Acknowledgements

A significant amount of the text in Section 3 was lifted from [RFC4642].

The author would like to acknowledge David Harrington, Miao Fuyou, Eric Rescorla, Juergen Schoenwaelder, Simon Josefsson, Olivier Coupelon, Alfred Hoenes, and the NETCONF mailing list members for their comments on the document. The author also appreciates Bert Wijnen, Mehmet Ersue, and Dan Romascanu for their efforts on issues resolving discussion; and Charlie Kaufman, Pasi Eronen, and Tim Polk for the thorough review of previous versions of this document.

7. Contributor's Address

Ibrahim Hajjeh
Ineovation
France

EMail: ibrahim.hajjeh@ineovation.fr

Alan Luchuk
SNMP Research, Inc.
3001 Kimberlin Heights Road
Knoxville, TN 37920-9716

EMail: luchuk@snmp.com

Juergen Schoenwaelder
Jacobs University Bremen
Campus Ring 1
28725 Bremen
Germany

EMail: j.schoenwaelder@jacobs-university.de

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, March 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.
- [RFC6353] Hardaker, W., "Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)", RFC 6353, July 2011.

8.2. Informative References

- [RFC4642] Murchison, K., Vinocur, J., and C. Newman, "Using Transport Layer Security (TLS) with Network News Transfer Protocol (NNTP)", RFC 4642, October 2006.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event Notifications", RFC 5277, July 2008.
- [RFC5539] Badra, M., "NETCONF over Transport Layer Security (TLS)", RFC 5539, May 2009.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Appendix A. Change Log (to be removed by RFC Editor before publication)

A.1. From -01 to -02

- o Update the server identity check and add rfc6125
- o Update the closure session and remove text related to close_notify alert specified by TLS
- o Update YANG Module

A.2. From -00 to -01

- o Move RFC5539 to informative references and remove RFC4742.
- o Shorten the YANG object names;
- o Extend the YANG module to support configuration of PSK;

A.3. From RFC5539 to draft-badra-netconf-rfc5539bis-00

- o Added text on how the generation of a NETCONF username is done.
- o Added text on how does this document fulfill the requirements in 6241 for the format of the username.
- o Removed unneeded wording about client/server, and changed use of client/server, manager/agent to TLS client/server and NETCONF client/server.
- o Added text to Security Considerations about EOM issues.
- o Added option for the chunked encoding described in RFC6242.
- o Added 1.1 capability to enable the chunked encoding described in RFC6242.

Author's Address

Mohamad Badra
Dhofar University

Email: mbadra@gmail.com

NETCONF Working Group
Internet-Draft
Intended status: Experimental
Expires: April 20, 2013

T. Iijima
Hitachi, Ltd.
H. Kimura
Y. Atarashi
H. Higuchi
Alaxala Networks Corp.
Oct 17, 2012

NETCONF over WebSocket
draft-iijima-netconf-websocket-ps-04

Abstract

This memo proposes a way of transporting NETCONF over WebSocket protocol. Web-related technologies are advancing and number of Web-based systems are increasing. Management systems that adopt Web-related technologies or that have browser-based management interface are getting common. It is natural to expect that there is a standard network operation and management protocol to be used over HTTP. Currently, however, there are few efforts in this area. Although NETCONF[RFC6241] once defined itself to be sent over SOAP/HTTPS[RFC4743], supposedly due to unfamiliarity to SOAP or lack of bi-directional capability, such efforts aren't successful enough[I-D.ietf-netconf-rfc4743-rfc4744-to-historic]. But now, WebSocket protocol, the update of HTTP equipped with bi-directional capability, is available[RFC6455]. This memo describes how NETCONF should be treated over WebSocket protocol.

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 April 20, 2013.

Copyright Notice

Copyright (c) 2012 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
2. Problem Statement	4
3. Use Case	6
4. Concerns about Using HTTP and WebSocket	7
5. Handling of NETCONF Username	8
6. Transporting NETCONF Messages over WebSocket Protocol	9
6.1. Message Sequence	9
6.2. WebSocket Message at Handshake from NETCONF Client	11
6.3. WebSocket Message at Handshake from NETCONF Server	12
6.4. NETCONF Message over WebSocket at NETCONF Client	12
6.5. NETCONF Message over WebSocket at NETCONF Server	14
7. Security Considerations	15
8. IANA Considerations	16
9. Acknowledgements	17
10. References	18
10.1. Normative References	18
10.2. Informative References	18
Authors' Addresses	20

1. Introduction

Web-related technologies are advancing and number of Web-based systems are increasing. Management systems that adopt Web-related technologies or have browser-based management interface are getting common. It is natural to expect that there is a standard network operation and management protocol to be sent over HTTP. Currently, however, there are few efforts in this area and, in most cases, management interface to be used over HTTP are proprietary. Although NETCONF[RFC6241] once defined itself to be sent over SOAP/HTTPS[RFC4743], supposedly due to unfamiliarity to SOAP or lack of bi-directional capability, such efforts didn't succeed well enough[I-D.ietf-netconf-rfc4743-rfc4744-to-historic]. But now, WebSocket protocol, the update of HTTP equipped with bi-directional capability, is available[RFC6455]. This memo describes how NETCONF should be exchanged over WebSocket protocol.

This memo does not intend to make WebSocket as a mandatory transport protocol for NETCONF. NETCONF specifies that it is mandatory to be sent over SSH[RFC6241]. But [RFC6241] also specifies in its section 2 that "the NETCONF protocol can be layered on any transport protocol that provides the required set of functionality." According to the specification, those required set of functionality are "Connection-Oriented Operation" and "Authentication, Integrity, and Confidentiality." WebSocket protocol meets those requirements. It is 'connection-oriented.' And, as written in the section 10.5 of [RFC6455], 'authentication' is ensured by mechanisms available to a generic HTTP server, such as cookies, HTTP Authentication, or TLS. Moreover, as written in the section 10.6 of [RFC6455], 'integrity and confidentiality' are ensured by running WebSocket protocol over TLS. For these reasons, WebSocket protocol coupled with TLS meets the requirements of transport protocol for NETCONF.

2. Problem Statement

Web-related technologies, such as JavaScript and JSON(JavaScript Orient Notation), are widely used. And number of Web-based systems, such as those provided for IaaS (Infrastructure as a Service), are increasing. There are systems that are providing management interface in a form of REST API(Application Programming Interface). It is natural to expect that there is a standard network operation and management protocol to be sent over HTTP. Currently, however, there are few efforts in this area. And, in most of the cases, management interface are proprietary.

NETCONF[RFC6241] once defined itself to be sent over SOAP/HTTPS[RFC4743], as drawn in Figure 1. But, supposedly due to unfamiliarity to SOAP or lack of bi-directional capability, such efforts haven't been successful enough[I-D.ietf-netconf-rfc4743-rfc4744-to-historic].

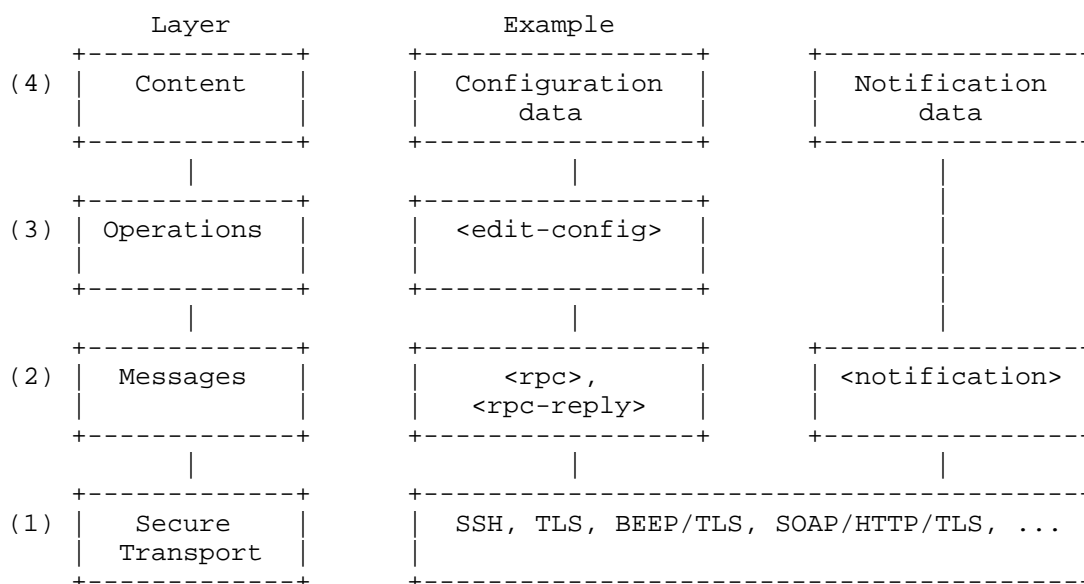


Figure 1: NETCONF Protocol Layers

As of now, however, WebSocket protocol is available[RFC6455]. It is based on HTTP, which is familiar to all. And, it has a bi-directional capability. Thus, by using WebSocket protocol, it's possible to realize NETCONF that is used over HTTP with ease and that supports notification mechanism specified in [RFC5277].

Some WebSocket implementations already exist today. As examples of WebSocket server implementation, Jetty[Jetty], Kaazing[Kaazing], and the like are available. And, as examples of WebSocket client implementation, Jetty[Jetty] is available. Moreover, as examples of Web-browsers that can work as WebSocket client, Chrome[Chrome], and FireFox[FireFox] and the like are available.

WebSocket server and client implementations mentioned above are providing libraries. And WebSocket protocol itself also defines API[WebSocket API] to be used on Web-browsers. Thus, by using those libraries and APIs, developers can develop NETCONF server, install-based NETCONF client, and browser-based NETCONF client with ease that use WebSocket for transporting NETCONF.

3. Use Case

XML, which NETCONF uses for message encoding, has high compatibility with HTTP in that XML are easily manipulated on Web-browsers by JavaScript DOM (Document Object Model) API. Hence, there are cases in which XML is used over HTTP to manage network devices. But as far as those XML are proprietary, the way of managing network devices and items to manage are different from network device to network device. If NETCONF, instead of proprietary XML, and its data models are used for above cases, it will provide a way of managing various network devices in a same manner.

Browser-based network management systems don't require installation on computers. For this reason, some operators prefer browser-based network management systems. This trend might accelerate in the age of tablet computers. In this respect, it is rational for NETCONF to have an option to be used through browser-based network management system. Operators will be able to manage network devices from any computers without installation.

4. Concerns about Using HTTP and WebSocket

There are some drawbacks inherent in HTTP as mentioned in the section 2.4 of [RFC4743], which describes the way of transporting NETCONF over SOAP/HTTPS. But, for these drawbacks, the same section makes suggestions. Those suggestions are effective when WebSocket is used for transporting NETCONF. That is, intermediate proxies SHOULD not be used since it may close idle connections. And, the fields of 'Cache-Control' and 'Pragma' in HTTP header, which is sent before or during WebSocket opening handshake, SHOULD be specified as 'no-cache.'

WebSocket has had several security concerns. But it incorporated its own security mechanisms such as origin header and masking, as stated in the Security Considerations section of [RFC6455]. In any case, using TLS is necessary for ensuring authentication and confidentiality, when WebSocket is used for transporting NETCONF.

5. Handling of NETCONF Username

NETCONF[RFC6241] mandates underlying transport protocol to carry NETCONF username and to provide it to NETCONF server. In the case of transporting NETCONF over WebSocket, this memo proposes that NETCONF username SHOULD be carried and provided in compliance with NETCONF over TLS[I-D.badra-netconf-rfc5539bis].

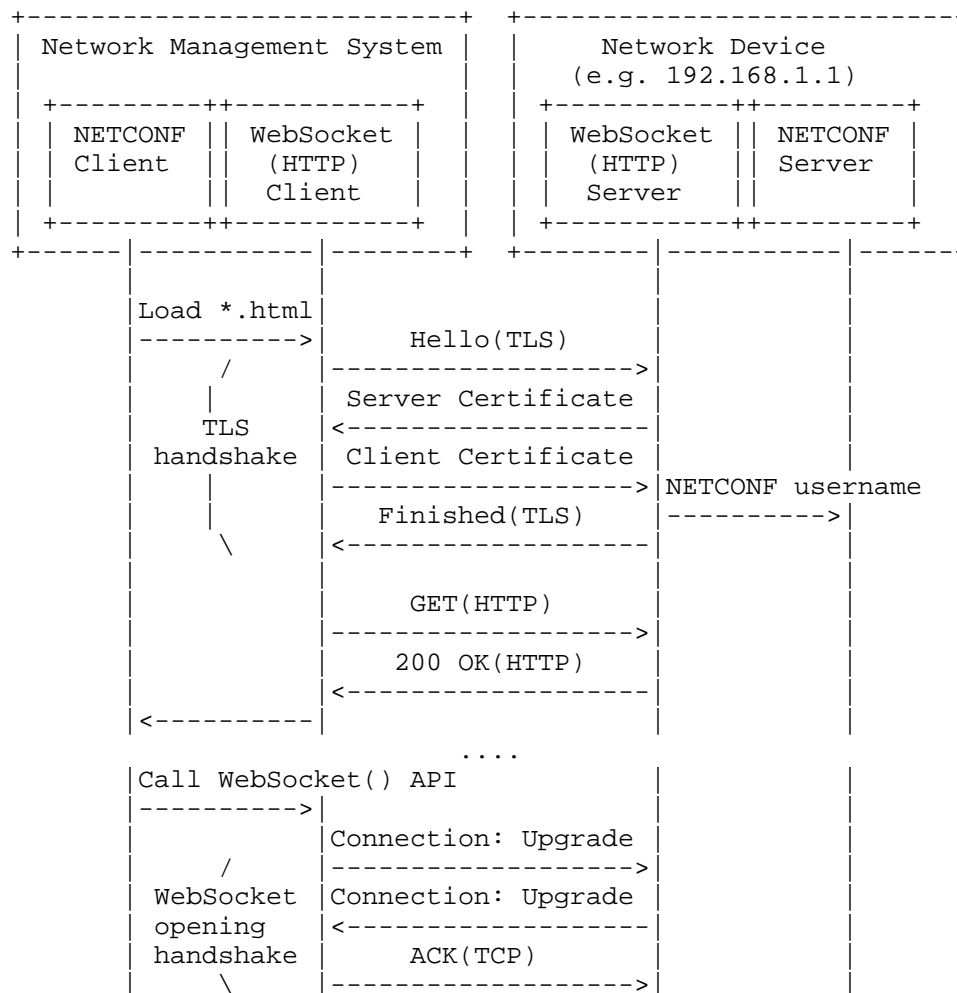
In the case of transporting NETCONF over WebSocket, TLS is necessary for the user authentication. In order to ensure that NETCONF access control is done consistently with TLS authentication, NETCONF username SHOULD be matched with TLS authentication data. At present, [I-D.badra-netconf-rfc5539bis] is proposing ways of extracting NETCONF username from TLS authentication data. Being compliant with these approaches is the most efficient. At least, it is confirmed that NETCONF server using WebSocket/TLS for underlying protocol can see TLS Certificate at the time of TLS handshake and can extract NETCONF username from the Certificate.

6. Transporting NETCONF Messages over WebSocket Protocol

This section specifies how NETCONF messages are exchanged between NETCONF client and server over WebSocket protocol.

6.1. Message Sequence

Simplified overall message sequence is depicted in Figure 2. This sequence is depicting the case in which NETCONF client runs on a Web-browser. However, it must be noted that there is also a case in which NETCONF client runs as an install-based software developed with WebSocket client library.



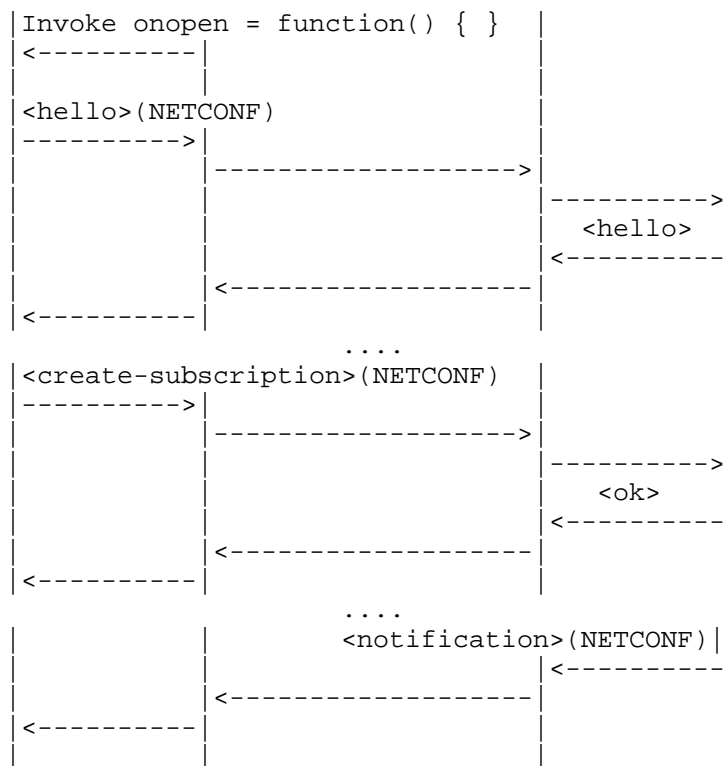


Figure 2: Message Sequence

First of all, a browser starts loading of an html file, which imports the code of NETCONF client, over TLS. This invokes TLS handshake. At the time of TLS handshake, NETCONF server can extract NETCONF username from Certificate according to the algorithm described in [I-D.badra-netconf-rfc5539bis]. After TLS handshake is complete, the html file is loaded onto the browser.

The loaded html file works as NETCONF client and it initiates WebSocket opening handshake. When NETCONF server receives a WebSocket connection request, it notifies the client of whether WebSocket handshake has succeeded. After WebSocket connection is established, NETCONF client starts sending NETCONF messages over the connection.

NETCONF <hello> messages are exchanged between NETCONF client and server, at first, so that a NETCONF session is established and a NETCONF session ID is allocated by NETCONF server to the NETCONF client. Then, NETCONF <rpc> messages are exchanged. After NETCONF

<rpc> message of <create-subscription> request is approved by the NETCONF server, NETCONF <notification> messages are sent from NETCONF server asynchronously.

When WebSocket server shuts down, NETCONF session as well as WebSocket connection is killed. Since NETCONF notification subscription is associated with NETCONF session ID as written in section 3.5 of NETCONF notification mechanism[RFC5277], subscription status is lost when WebSocket server shuts down. Thus, it is necessary to redo WebSocket opening handshake, NETCONF session establishment, and NETCONF notification subscription after WebSocket server reboots.

Without any indications, TCP port numbers of 443 is automatically used for transporting NETCONF messages over WebSocket over TLS. When port number other than 443 needs to be used, the number SHOULD be specified at both NETCONF client and server respectively.

6.2. WebSocket Message at Handshake from NETCONF Client

WebSocket opening handshake is necessary for the establishment of an WebSocket connection, which is used for NETCONF message exchange. The WebSocket handshake is initiated by a NETCONF client. Figure 3 is an example of WebSocket message sent from the NETCONF client at the time of WebSocket handshake.

```
C: GET /netconf HTTP/1.1
C: Host: 192.168.1.1
C: Upgrade: websocket
C: Connection: Upgrade
C: Sec-WebSocket-Key: dGh1IHNhbXBsZSBub25jZQ==
C: Origin: http://192.168.1.1
C: Sec-WebSocket-Protocol: netconf
C: Sec-WebSocket-Version: 13
```

Figure 3: WebSocket Message from NETCONF Client

Most of the fields in Figure 3 are generated automatically by WebSocket client. The only field that the NETCONF client needs to specify is 'Sec-WebSocket-Protocol' field, so-called subprotocol field. The NETCONF client has to specify the field as 'netconf,' for example, so that the NETCONF server understands that messages sent over this connection is directed towards NETCONF server.

Aforementioned establishment of the WebSocket connection and specification of subprotocol is made by using WebSocket API in the

html file of the NETCONF client as depicted in Figure 4.

```
var wssURL = "wss://" + location.host;  
var wss = new WebSocket(wssURL, "netconf");
```

Figure 4: WebSocket API in NETCONF Clients for Initiating Handshake

6.3. WebSocket Message at Handshake from NETCONF Server

Figure 5 is an example of WebSocket message sent from the NETCONF server to the NETCONF client at the time of WebSocket handshake.

```
S: HTTP/1.1 101 Switching Protocols  
S: Upgrade: websocket  
S: Connection: Upgrade  
S: Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=  
S: Sec-WebSocket-Protocol: netconf
```

Figure 5: WebSocket Message from NETCONF Server

Most of the fields in Figure 5 are generated automatically by WebSocket server, too. The only field that the NETCONF server needs to specify is 'Sec-WebSocket-Protocol' field. The NETCONF server has to specify the field as 'netconf,' for example, in order to let the NETCONF client know that the WebSocket server in the network device accepts NETCONF messages.

Unlike the WebSocket client, there's no standardized WebSocket API for WebSocket server. But, there are some WebSocket server implementations as mentioned in section 2. Thus, it's possible to develop a part of WebSocket opening handshake at NETCONF server with ease by using libraries provided by those implementations.

6.4. NETCONF Message over WebSocket at NETCONF Client

NETCONF message exchange between NETCONF client and server starts after an WebSocket connection is established. <hello> messages are exchanged, first, for the establishment of a NETCONF session and allocation of NETCONF session ID. Then, <rpc> messages like <edit-config> are sent from NETCONF client to NETCONF server for NETCONF configurations. And <rpc> messages like <create-subscription> are sent from NETCONF client to NETCONF server for subscription of NETCONF notification. Moreover, messages like <notification> are sent asynchronously from NETCONF server to NETCONF client for NETCONF

notification. During this data transfer period, both NETCONF configuration messages and NETCONF notification messages are encapsulated as a payload of WebSocket protocol according to the Data Framing specified in the section 5.2 of WebSocket protocol[RFC6455]. This encapsulation is made by WebSocket layer.

Sending and receiving of NETCONF messages at NETCONF client is made by using WebSocket API. The example is illustrated in Figure 6.

```
wss.onopen = function() {  
    // WebSocket connection has been established.  
    // Thus, NETCONF <hello> can be sent here  
    // by send() method.  
    wss.send("message to send");  
    ....  
};  
  
wss.onmessage = function (evt) {  
    // NETCONF message has arrived.  
    // Thus, NETCONF message can be parsed here by DOM APIs.  
  
    var parser = new DOMParser();  
    var dom     = parser.parseFromString(evt.data, "text/xml");  
    ....  
    if(dom.documentElement.nodeName == "hello"){  
        // The NETCONF message turned out to be  
        // NETCONF <hello>.  
        // Subsequent NETCONF message can be sent here  
        // by send() method.  
        wss.send("message to send");  
        ...  
    }  
};
```

Figure 6: WebSocket API in NETCONF Client for Sending Messages

As shown in Figure 6, NETCONF messages are sent from NETCONF client by using WebSocket API of "wss.send()." And, NETCONF messages are received at NETCONF client by using WebSocket API of "wss.onmessage()."

The contents of NETCONF messages exchanged through above API might be either a hand-written XML messages typed into network management system or XML messages created by JavaScript DOM API according to the data set on the network management system. It must be noted that in the case of transporting NETCONF over WebSocket, <rpc> parts need to

be created by NETCONF client, unlike the case of transporting NETCONF over SOAP/HTTPS, in which <rpc> parts are generated in SOAP layer.

6.5. NETCONF Message over WebSocket at NETCONF Server

Unlike the WebSocket client, the way of sending and receiving NETCONF message at NETCONF server is proprietary since there's no standardized API for WebSocket server. But, there are some WebSocket server implementations as mentioned in section 2. Thus, it's possible to develop NETCONF server with ease by using libraries provided by those implementations.

7. Security Considerations

It is necessary to use TLS (Transport Layer Security) in order to ensure Transport-level security, such as authentication of users and encryption of data transfer. That is, NETCONF has to be sent in the form of NETCONF over WebSocket over TLS (WSS).

In addition, the security considerations of NETCONF protocol[RFC6241], NETCONF Notification mechanism[RFC5277], and WebSocket protocol[RFC6455] are applicable to this document. Implementers or users SHOULD take these considerations into account.

8. IANA Considerations

As written in section 11.5 of WebSokcet protocol[RFC6455], NETCONF's Subprotocol Common Name, to be exchanged in 'Sec-WebSocket-Protocol' field at WebSocket opening handshake, coupled with Identifier and Definition need to be registered with the WebSocket Subprotocol Name Registry.

9. Acknowledgements

This document was written using the xml2rfc tool described in [RFC2629].

10. References

10.1. Normative References

- [I-D.badra-netconf-rfc5539bis]
Badra, M., "NETCONF Over Transport Layer Security (TLS)",
draft-badra-netconf-rfc5539bis-02 (work in progress),
April 2012.
- [I-D.ietf-netconf-rfc4743-rfc4744-to-historic]
Wijnen, B., "RFC4743 and RFC4744 to Historic status",
draft-ietf-netconf-rfc4743-rfc4744-to-historic-00 (work in
progress), September 2012.
- [RFC4743] Goddard, T., "Using NETCONF over the Simple Object
Access Protocol (SOAP)", RFC 4743, December 2006.
- [RFC5277] Chisholm, S. and H. Trevino, "NETCONF Event
Notifications", RFC 5277, July 2008.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A.
Bierman, "Network Configuration Protocol (NETCONF)",
RFC 6241, June 2011.
- [RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol",
RFC 6455, December 2011.
- [WebSocket API]
"The WebSocket API".

<<http://dev.w3.org/html5/websockets/>>

10.2. Informative References

- [Chrome] "Chrome".

<<http://www.google.com/chrome/?hl=en>>
- [FireFox] "FireFox".

<<http://www.mozilla.org/>>
- [Jetty] "Jetty WebServer".

<<http://jetty.codehaus.org/jetty/>>
- [Kaazing] "Kaazing".

<<http://kaazing.com/>>

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
June 1999.

Authors' Addresses

Tomoyuki Iijima
Hitachi, Ltd.
292 Yoshida-cho, Totsuka-ku
Yokohama, Kanagawa 244-0817
Japan

Phone: +81-45-860-2156
Email: tomoyuki.iijima.fg@hitachi.com

Hiroyasu Kimura
Alaxala Networks Corp.
Shin-Kawasaki Mitsui Bldg.
890 Saiwai-ku Kashimada
Kawasaki, Kanagawa 212-0058
Japan

Phone: +81-44-549-1735
Fax: +81-44-549-1272
Email: h-kimura@alaxala.net

Yoshifumi Atarashi
Alaxala Networks Corp.
Shin-Kawasaki Mitsui Bldg.
890 Saiwai-ku Kashimada
Kawasaki, Kanagawa 212-0058
Japan

Phone: +81-44-549-1735
Fax: +81-44-549-1272
Email: atarashi@alaxala.net

Hidemitsu Higuchi
Alaxala Networks Corp.
Shin-Kawasaki Mitsui Bldg.
890 Saiwai-ku Kashimada
Kawasaki, Kanagawa 212-0058
Japan

Phone: +81-44-549-1735
Fax: +81-44-549-1272
Email: hidemitsu.higuchi@alaxala.com

This Internet-Draft, draft-schoenw-netconf-light-00.txt, has expired, and has been deleted from the Internet-Drafts directory. An Internet-Draft expires 185 days from the date that it is posted unless it is replaced by an updated version, or the Secretariat has been notified that the document is under official review by the IESG or has been passed to the RFC Editor for review and/or publication as an RFC. This Internet-Draft was not published as an RFC.

Internet-Drafts are not archival documents, and copies of Internet-Drafts that have been deleted from the directory are not available. The Secretariat does not have any information regarding the future plans of the authors or working group, if applicable, with respect to this deleted Internet-Draft. For more information, or to request a copy of the document, please contact the authors directly.

Draft Authors:

Vladislav Perelman<v.perelman@jacobs-university.de>

Juergen Schoenwaelder<j.schoenwaelder@jacobs-university.de>

Mehmet Ersue<mehmet.ersue@nsn.com>