NETCONF Working Group
Internet-Draft
Intended status: Experimental
Expires: April 23, 2012

T. Iijima
Hitachi, Ltd.
H. Kimura
Y. Atarashi
H. Higuchi
Alaxala Networks Corp.
Oct 21, 2011

NETCONF over WebSocket
draft-iijima-netconf-websocket-ps-01

Abstract

This memo proposes transporting NETCONF over WebSocket protocol[I-D.ietf-hybi-thewebsocketprotocol]. The use of HTTP and Web-based applications are increasing with the emergence of advanced drawing technologies and cloud computing. IT management systems supporting browser-based interface are getting common. It's natural for network management systems to support Web-based interface. Currently, however, few network management protocols support the transportation over HTTP. Although NETCONF[RFC6241] was defined to be sent over SOAP/HTTPS[RFC4743], it was excluded from the options of realizing notification mechanism[RFC5277] since HTTP lacks bi-directional capabilty. At present, WebSocket protocol, the update of HTTP with bi-directional capability, is available. This memo describes the way NETCONF is sent 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 21, 2012.

Copyright Notice

Copyright (c) 2011 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

The use of HTTP and Web-based applications are increasing in contrast with install-based applications with the emergence of advanced drawing technologies and cloud computing. IT management systems supporting browser-based interface to control IT resources are getting common. It's natural for network management systems to support Web-based interface. Currently, however, few network management protocols support transportation over HTTP. Although NETCONF[RFC6241] was defined to be sent over SOAP/HTTPS[RFC4743], it was excluded from the options of realizing notification mechanism[RFC5277] since HTTP lacks bi-directional capabilty. At present, WebSocket protocol, the update of HTTP with bi-directional capability, is available[I-D.ietf-hybi-thewebsocketprotocol]. This memo describes the way NETCONF is sent over WebSocket protocol.

This memo does not intend to standardize NETCONF over WebSocket as a mandatory transport mapping of NETCONF. NETCONF over SSH is defined to be mandatory in RFC6241[RFC6241]. But section 2 of RFC6241 also specifies that "the NETCONF protocol can be layered on any transport protocol that provides the required set of functionality." According to the description of those requirements, "Connection-Oriented Operation" and "Authentication, Integrity, and Confidentiality" are the required set of functionality. WebSocket protocol meets those requirements. It is connection-oriented. And, authentication, integrity, and confidentiality are ensured by technologies such as masking of payload and sending over TLS. Thus, WebSocket can become one of the optional protocols to transport NETCONF.

2. Problem Statement

Figure 1 is an architecture of NETCONF extracted from RFC6241[RFC6241]. So far there's few ways to easily provide browser-based network management system using NETCONF. Even though SOAP/HTTP/TLS is defined as one of the transport protocols for NETCONF, it can not provide notification mechanism since HTTP lacks bi-directional capability. AJAX and Comet technologies are often used to outwardly provide HTTP with bi-directinal capability by way of polling or keeping a long session at the application level. But, those technologies tend to consume large amount of bandwidth and memories. There are cases where such technologies can not be accomodated into network devices due to their lack of resources (memory, processor).

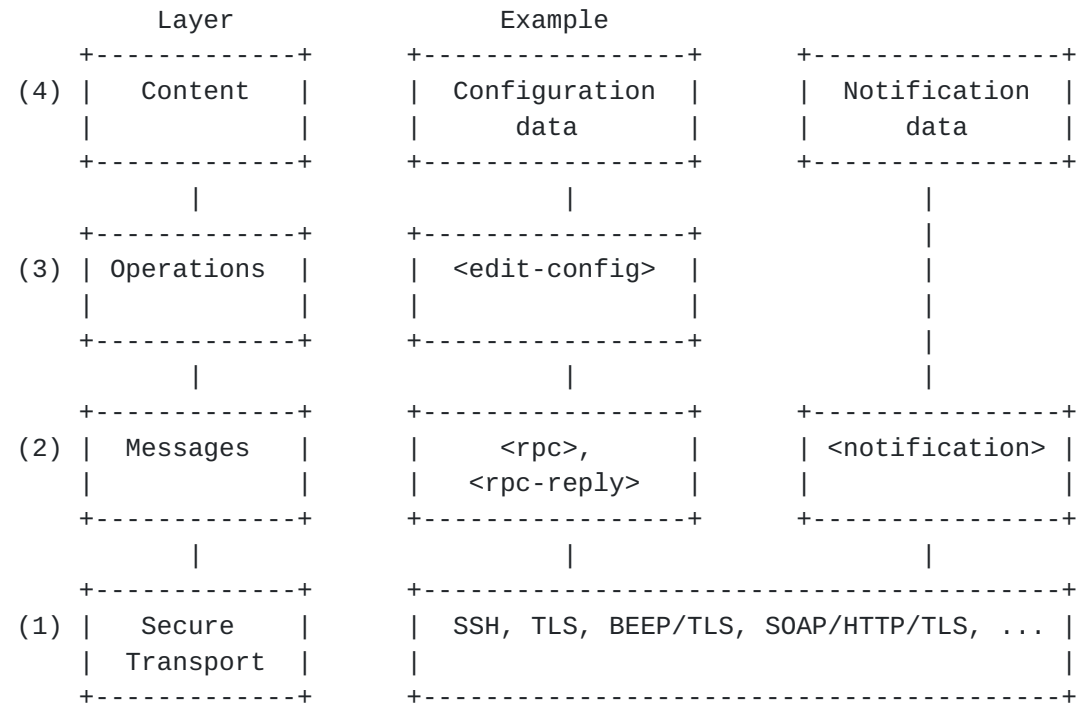

Figure 1: NETCONF Protocol Layers

At present, WebSocket protocol is being standardized. It is based on HTTP and has a bi-directional capability. Although standardization is still ongoing in IETF, there are already some implementations. Jetty[Jetty], Kaazing[Kaazing], and Apache/pywebsocket are available as WebSocket servers. And, Chrome[Chrome] and FireFox[FireFox] are available as browsers to run WebSocket clients. By using WebSocket as an underlying protocol for NETCONF, NETCONF client in a network management system become browser-based. And, realtime event

notification is possible. This is beneficial for a network management system that provides failure detection function.

3. Use Case

Some vendors are providing network management systems that put XML messages directly onto HTTP. In some cases, API is provided in a style of XML, and operators write XML messages by hands and send those messages through network management system. It's not scalable, but one of the advantages of this style is that it does not require much programming. However, as far as the parts of XML is proprietary, those network management systems can manage limited number of network devices. NETCONF and its data models can be used for this purpose.

And, WebSocket is defining JavaScript API[WebSocket API], which is supported by current Web browsers. By using WebSocket API to exchange NETCONF messages, NETCONF client becomes browser-based. Browser-based network management systems don't require installation in computers. Thus, they can be provided without much care about the platform on which network management systems run. For this reason, they can be used even by tablet computers, which will be deployed widely in a cloud computing age.

In addition, there are various APIs to control computer and storage resources, which are also intended to be used by Web browser. DMTF (Distributed Management Task Force), for example, is standardizing REST-based API in CMWG (Cloud Management Working Group) for controlling computer and storage resources[DMTF]. REST-based API is easily manipulated by JavaScript language. And, SNIA (Storage Networking Industry Association) defines CDMI (Cloud Data Management Interface), a cloud strage API[CDMI]. This is also manipulated by JavaScript language. By using WebSocket API for NETCONF notification and combining these APIs, it's possible to develop management systems that can control IT resources according to situations in networks.

4. Concerns about HTTP and WebSocket

HTTP and WebSocket, on the other hand, have several concerns to pay attention to. As Security Considerations section of [I-D.ietf-hybi-thewebsocketprotocol] mentions, HTTP and WebSocket have quite a few risks of being attacked for its wide usage. But, as stated in the same section of [I-D.ietf-hybi-thewebsocketprotocol], WebSocket incorporates its own security mechanisms such as challenge-response exchange and payload masking. In addition, TLS is used when strong care needs to be paid on information transported over WebSocket protocol.

Section 2.4 of RFC 4743[RFC4743] also mentions some drawbacks inherent in HTTP. But the section makes suggestion offsetting those drawbacks. 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 field of 'Cache-Control' and 'Pragma' in HTTP header sent before and during WebSocket handshake should be specified as 'no-cache,' as in the case of transporting NETCONF over SOAP/HTTPS.

5. Messages of NETCONF over WebSocket

This section specifies the messages exchanged between an NETCONF client and an NETCONF server when WebSocket protocol is used for transporting NETCONF.

5.1. Message Sequence

Overall message sequence is depicted in Figure 2.

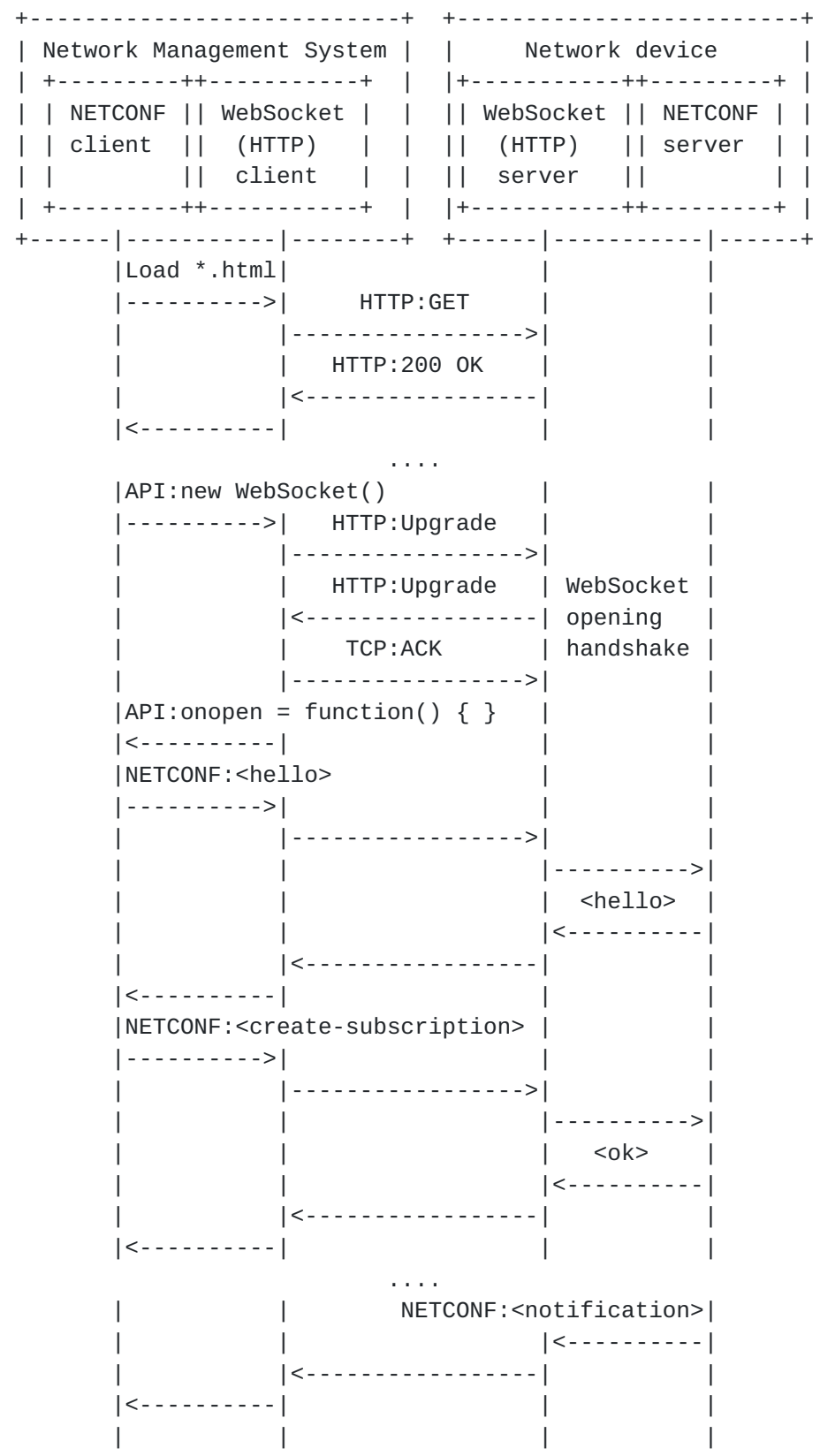
Network Management System
NETCONF client
WebSocket (HTTP) client
Network device
WebSocket (HTTP) server
NETCONF server
Load *.html
HTTP:GET
HTTP:200 OK
....
API:new WebSocket()
HTTP:Upgrade
HTTP:Upgrade
WebSocket opening handshake
TCP:ACK
API:onopen = function() { }
NETCONF:<hello>
<hello>
NETCONF:<create-subscription>
<ok>
....
NETCONF:<notification>

Figure 2: Message Sequence

When a browser loads an html file which imports the code of NETCONF client using WebSocket API, it initiates WebSocket opening handshake. After NETCONF client is notified of the success of WebSocket handshake, it starts sending NETCONF messages. NETCONF <rpc> messages are exchanged, after an NETCONF session is established and an NETCONF session ID is allocated by the NETCONF server to the NETCONF client. NETCONF <notification> messages are sent from NETCONF server, after an NETCONF session ID is allocated and subscription request is approved by the NETCONF server.

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 [RFC5277], status of subscription is lost when WebSocket server shuts down. Thus, WebSocket opening handshake, NETCONF session establishment, and NETCONF notification subscription need to be made again after WebSocket server reboots.

TCP port numbers of 80 and 443 can be used for transporting NETCONF messages over WebSocket and for transporting NETCONF messages over WebSocket over TLS, respectively. In this case, indication of TCP port numbers in NETCONF client is unnecessary. When any TCP port numbers other than official HTTP and HTTPS numbers are assigned for this purpose, indication of port numbers in NETCONF client as well as NETCONF server is necessary. And, setting about firewall should be opened for these port numbers.

5.2. WebSocket Message from NETCONF Client at WebSocket Handshake

WebSocket opening handshake is necessary before an NETCONF client establishes an NETCONF session. The WebSocket handshake is initiated by the 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: netconfdevice
C: Upgrade: websocket
C: Connection: Upgrade
C: Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
C: Origin: http://example.com
C: Sec-WebSocket-Protocol: netconf
C: Sec-WebSocket-Version: 13
```

Figure 3: WebSocket Message from NETCONF Client

As written in Figure 3, GET instead of POST is specified in the WebSocket header. Fields of 'Upgrade' and 'Connection' are specified as 'websocket' and 'Upgrade' respectively, so that an HTTP server running in the NETCONF server understands that it needs to work as an WebSocket server. Challenge-response mechanism encoded in base64 is included in the 'Sec-WebSocket-Key' field. And, Origin field needs to be validated by the WebSocket server. But, these are generated automatically by WebSocket client. The field that the NETCONF client needs to specify is that of 'Sec-WebSocket-Protocol,' so-called subprotocol field. The NETCONF client has to specify it, for example, as 'netconf,' so that the WebSocket server understands that it needs to hand over messages sent over this connection to the NETCONF server.

Aforementioned establishment of the WebSocket connection and specification of subprotocol is made by using WebSocket API as depicted in Figure 4 in the NETCONF clients.

```
var wsURL = "ws://" + host;
ws = new WebSocket(wsURL, "netconf");
```

Figure 4: WebSocket API in NETCONF Clients for Initiating Handshake

5.3. WebSocket Message from NETCONF Server at WebSocket Handshake

Figure 5 is an example of WebSocket message sent from an NETCONF server to an 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

Unlike the WebSocket client, there's no standardized WebSocket API for WebSocket server. Although the way of implementing WebSocket server is proprietary, there's some implementations as mentioned in section 2. NETCONF server can be develped on top of those implementations by using their libraries.

As written in Figure 5, when HTTP server inside network device can run as WebSocket server, the number of 101 has to be returned to

WebSocket client. Fields of 'Upgrade' and 'Connection' are specified as 'websocket' and 'Upgrade' respectively, in order to let the NETCONF client know that an HTTP server in network device can run as WebSocket server. In this case too, above fields need to be generated by WebSocket server. The field that the NETCONF server needs to specify is that of 'Sec-WebSocket-Protocol.' The NETCONF server has to specify its name of the subprotocol, for example, as 'netconf,' in order to let the NETCONF client know that the WebSocket server in the network device accepts NETCONF messages.

5.4. NETCONF Message over WebSocket Connection

After an WebSocket connection is established, an NETCONF client and an NETCONF server start exchanging NETCONF messages. Exchanges of <hello> messages between NETCONF client and NETCONF server start from this timing. After an NETCONF session is established and a session ID is allocated to NETCONF client, <rpc> messages containing <edit-config> for NETCONF configurations and <rpc> messages like <create-subscription> for NETCONF notification should be sent from the NETCONF client to the NETCONF server. And messages like <notification> should be sent from the NETCONF server to the NETCONF client asynchronously. During this data transfer period, so called HTTP headers coupled with GET, 404 and so on are unnecessary. NETCONF configuration messages as well as NETCONF notification messages are encapsulated as a payload of WebSocket protocol according to the Data Framing specified in the section 5.2 of [I-D.ietf-hybi-thewebsocketprotocol]. And above encapusulated data are exchanged over TCP without the use of HTTP.

Sending of NETCONF message from NETCONF client is made by using WebSocket API as depicted in Figure 6.

```
ws.onopen = function() {
  // WebSocket connection has established.
  // rpc message of <hello> can be sent here
  // by send() method.
  ws.send("message to send");
  ....
};

ws.onmessage = function (evt) {
  // NETCONF message has arrived.
  // NETCONF message can be parsed by DOM APIs.
  var received_msg = evt.data;

  var parser = new DOMParser();
  var dom    = parser.parseFromString(evt.data, "text/xml");
  if(dom.documentElement.nodeName == "hello"){
    // rpc reply message of <hello> has arrived.
    // Subsequent NETCONF message can be sent here
    // by send() method.
    ws.send("message to send");
    ...
  }
};
```

Figure 6: WebSocket API in NETCONF Client for Sending Messages

As shown in Figure 6, NETCONF messages are sent from WebSocket API. The NETCONF message to send through WebSocket API can be a hand-written message typed in from browser-based NMS. Or it might be created by JavaScript DOM API according to the data typed in from browser-based NMS. Unlike the case of NETCONF over SOAP over HTTPS, which provides API to manipulate NETCONF <rpc>, parts of <rpc> as well as contents need to be manipulated by NETCONF client.

NETCONF messages are also received by WebSocket API. These messages are analyzed by JavaScript DOM API.

In contrast, the way of sending NETCONF message from NETCONF server is proprietary. Unlike the WebSocket client, there's no standardized WebSocket API for WebSocket server. Although the way of implementing WebSocket server is proprietary, there's some implementations as mentioned in section 2. NETCONF server can be developed on top of those implementations by using their libraries. By using send method of those libraries, it's possible to send NETCONF notification message asynchronously from NETCONF server to NETCONF clients.

6. Security Considerations

The security considerations of [RFC6241], [RFC5277], and [I-D.ietf-hybi-thewebsocketprotocol] are applicable to this document. Implementers or users should take these considerations into account.

Transport-level security, such as authentication of users and encryption of data transfer, has to be ensured by TLS (Transport Layer Security). That is, security has to be provided in the form of NETCONF over WebSocket over TLS (WSS).

7. IANA Considerations

As written in section 11.5 of [I-D.ietf-hybi-thewebsocketprotocol], registration of NETCONF's Identifier, to be exchanged at WebSocket opening handshake, and Common Name with WebSocket Subprotocol Name Registry is necessary.

8. Acknowledgements

This document was written using the xml2rfc tool described in RFC 2629 [RFC2629].

9. References

9.1. Normative References

[I-D.ietf-hybi-thewebsocketprotocol]
Fette, I. and A. Melnikov, "The WebSocket protocol", draft-ietf-hybi-thewebsocketprotocol-17 (work in progress), September 2011.

[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.

[WebSocket API]
"The WebSocket API".

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

9.2. Informative References

[CDMI] "SNIA's CDMI".

<http://cdmi.sniacloud.com/>

[Chrome] "Chrome".

<http://www.google.com/chrome/?hl=en>

[DMTF] "CLOUD | DMTF".

<http://www.dmtf.org/standards/cloud>

[FireFox] "FireFox".

<http://www.mozilla.org/>

[I-D.moffitt-xmpp-over-websocket]
Moffitt, J. and E. Cestari, "An XMPP Sub-protocol for WebSocket", draft-moffitt-xmpp-over-websocket-00 (work in progress), December 2010.

[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.com

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