Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2015

N. Wu
S. Zhuang
Huawei

March 6, 2015

A YANG Data Model for Flow Specification
draft-wu-rtgwg-flowspec-cfg-00

Abstract

This document defines a YANG data model for Flow Specification implementations. The data model includes configuration data and state data.

Requirements Language

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 RFC 2119 [RFC2119].

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 September 7, 2015.

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model for Flow Specification [RFC5575] implementations.

The data model convers configuration of filtering rules and actions of traffic flow specification. It also provides information about operational running state of traffic flow specification, which includes RIB of flow specification route, attributes dedicated to different protocols and statistics.

1.1. Terminology

The following terms are defined in [RFC6020]:

- configuration data
- data model
- module
- state data

The terminology for describing YANG data models is found in [RFC6020].

1.2. Tree Diagrams

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

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration data (read-write), and "ro" means 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. Flow Specification data model

The data model has the following structure for configuration of traffic flow specification:

```
+--rw flowspec-cfg
|  +--rw flow-route* [name]
|     +--rw name              string
|     +--rw filter-rules
|     |  +--rw destination-prefix
|     |  |  +--rw dest-prefix-length?   uint8
|     |  |  +--rw dest-prefix?          inet:ip-address
|     |  +--rw source-prefix
|     |  |  +--rw src-prefix-length?   uint8
|     |  |  +--rw src-prefix?          inet:ip-address
|     |  +--rw ip-protocol
|     |  |  +--rw op-protocol* [compare-protocol ip-proto]
|     |  |     +--rw compare-protocol    compare-ops
|     |  |     +--rw ip-proto            ip-protocol-types
|     |  +--rw port-number
|     |  |  +--rw op-port* [compare-port port]
|     |  |     +--rw compare-port    compare-ops
|     |  |     +--rw port            uint16
|     |  +--rw destination-port
|     |  |  +--rw op-destport* [compare-destport dest-port]
|     |  |     +--rw compare-destport    compare-ops
|     |  |     +--rw dest-port           uint16
|     |  +--rw source-port
```

```
|     |  |  +--rw op-srcport* [compare-srcport src-port]
|     |  |     +--rw compare-srcport    compare-ops
|     |  |     +--rw src-port           uint16
|     |  +--rw icmp-type-value
|     |  |  +--rw op-icmp-type* [compare-icmp-type icmp-type]
|     |  |     +--rw compare-icmp-type    compare-ops
|     |  |     +--rw icmp-type            uint8
|     |  +--rw icmp-code-value
|     |  |  +--rw op-code-value* [compare-icmp-code icmp-code]
|     |  |     +--rw compare-icmp-code    compare-ops
|     |  |     +--rw icmp-code            uint8
|     |  +--rw tcp-flag
|     |  |  +--rw op-bitmask* [tcp-flag-op tcp-flag-value]
|     |  |     +--rw tcp-flag-op       bitmask-ops
|     |  |     +--rw tcp-flag-value    uint16
|     |  +--rw pkt-len
|     |  |  +--rw op-pkt-len* [compare-pkt-len packet-length]
|     |  |     +--rw compare-pkt-len    compare-ops
|     |  |     +--rw packet-length      uint16
|     |  +--rw dscp
|     |  |  +--rw op-dscp* [compare-dscp dscp-value]
|     |  |     +--rw compare-dscp    compare-ops
|     |  |     +--rw dscp-value      dscp-type
|     |  +--rw fragment
|     |     +--rw op-fragment* [compare-fragment fragment-mode]
|     |        +--rw compare-fragment    bitmask-ops
|     |        +--rw fragment-mode       fragment-mode-type
|     +--rw filter-actions
|        +--rw traffic-rate
|        |  +--rw rate?   float
|        +--rw redirect
|        |  +--rw route-target?   string
|        +--rw traffic-marking
|        |  +--rw remark-dscp?   union
|        +--rw traffic-deny
|           +--rw deny?   boolean
```

This data model defines the configuration container for traffic flow specification. In this container, there is a list of configuration containers per flow specification route, which contains the configuration for filtering rules and corresponding actions.

The data model has the following structure for state of traffic flow specification:

```
+--ro flowspec-state
   +--ro flowspec-rib
   |  +--ro flowspec-route* [index islocal]
```

```
|     +--ro index                uint32
|     +--ro islocal              boolean
|     +--ro local-name?          string
|     +--ro neighbor?            inet:ip-address
|     +--ro duration?            uint32
|     +--ro filter-rules
|     |  +--ro destination-prefix
|     |  |  +--ro dest-prefix-length?   uint8
|     |  |  +--ro dest-prefix?          inet:ip-address
|     |  +--ro source-prefix
|     |  |  +--ro src-prefix-length?   uint8
|     |  |  +--ro src-prefix?          inet:ip-address
|     |  +--ro ip-protocol
|     |  |  +--ro op-protocol* [compare-protocol ip-proto]
|     |  |     +--ro compare-protocol    compare-ops
|     |  |     +--ro ip-proto            ip-protocol-types
|     |  +--ro port-number
|     |  |  +--ro op-port* [compare-port port]
|     |  |     +--ro compare-port    compare-ops
|     |  |     +--ro port            uint16
|     |  +--ro destination-port
|     |  |  +--ro op-destport* [compare-destport dest-port]
|     |  |     +--ro compare-destport    compare-ops
|     |  |     +--ro dest-port           uint16
|     |  +--ro source-port
|     |  |  +--ro op-srcport* [compare-srcport src-port]
|     |  |     +--ro compare-srcport    compare-ops
|     |  |     +--ro src-port           uint16
|     |  +--ro icmp-type-value
|     |  |  +--ro op-icmp-type* [compare-icmp-type icmp-type]
|     |  |     +--ro compare-icmp-type    compare-ops
|     |  |     +--ro icmp-type            uint8
|     |  +--ro icmp-code-value
|     |  |  +--ro op-code-value* [compare-icmp-code icmp-code]
|     |  |     +--ro compare-icmp-code    compare-ops
|     |  |     +--ro icmp-code            uint8
|     |  +--ro tcp-flag
|     |  |  +--ro op-bitmask* [tcp-flag-op tcp-flag-value]
|     |  |     +--ro tcp-flag-op       bitmask-ops
|     |  |     +--ro tcp-flag-value    uint16
|     |  +--ro pkt-len
|     |  |  +--ro op-pkt-len* [compare-pkt-len packet-length]
|     |  |     +--ro compare-pkt-len    compare-ops
|     |  |     +--ro packet-length      uint16
|     |  +--ro dscp
|     |  |  +--ro op-dscp* [compare-dscp dscp-value]
|     |  |     +--ro compare-dscp    compare-ops
|     |  |     +--ro dscp-value      dscp-type
```

```
|     |  +--ro fragment
|     |     +--ro op-fragment* [compare-fragment fragment-mode]
|     |        +--ro compare-fragment    bitmask-ops
|     |        +--ro fragment-mode       fragment-mode-type
|     +--ro filter-actions
|     |  +--ro traffic-rate
|     |  |  +--ro rate?   float
|     |  +--ro redirect
|     |  |  +--ro route-target?   string
|     |  +--ro traffic-marking
|     |  |  +--ro remark-dscp?   union
|     |  +--ro traffic-deny
|     |     +--ro deny?   boolean
|     +--ro protocol-specific
|        +--ro (protocol)?
|           +--:(bgp)
|           |  +--ro peer-rid?        inet:ipv4-address
|           |  +--ro isrefclient?     boolean
|           |  +--ro med?             uint32
|           |  +--ro preference?      uint32
|           |  +--ro local-pref?      uint8
|           |  +--ro origin?          enumeration
|           |  +--ro originator?      inet:ipv4-address
|           |  +--ro community?       string
|           |  +--ro ext-community?   string
|           |  +--ro cluster-list?    string
|           |  +--ro path-as?         string
|           +--:(ospf)
|           |  +--ro lsa-type?        string
|           |  +--ro ls-id?           inet:ipv4-address
|           |  +--ro advertiser?      inet:ipv4-address
|           |  +--ro area?            union
|           |  +--ro tag?             uint32
|           +--:(isis)
|              +--ro lsp-id?          string
|              +--ro priority?        uint32
|              +--ro admin-tag?       uint32
|
+--ro flowspec-statis
```

This data model defines two state containers for traffic flow specification. In the first container, there is a list of state containers per flow specification route, which contains the current state for filtering rules and actions. In the second container, there is statistics information for traffic flow specifications.

3. Flow Specification YANG Module

```
//<CODE BEGINS> file "ietf-flowspec@2015-02-27.yang"

module ietf-flowspec {
  namespace "urn:huawei:params:xml:ns:yang:flowspec";
  prefix flowspec;

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF RTGWG (Routing Working Group) Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>

     WG Chair: Chris Bowers
               <mailto:cbowers@juniper.net>

     WG Chair: Jeff Tantsura
               <mailto:jeff.tantsura@ericsson.com>

     Editor:   Shunwan Zhuang
               <mailto:zhuangshunwan@huawei.com>

     Editor:   Nan Wu
               <mailto:eric.wu@huawei.com>";


  description
    "This module contains a collection of YANG definitions for
     configuring flow specification implementations.

     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)."

  revision 2015-02-27 {
    description
      "Initial revision.";
    reference
      "draft-ietf-netmod-routing-cfg-16:
       A YANG Data Model for Routing Management";
  }

  /* Typedefs */
  typedef compare-ops {
    type enumeration {
      enum greater-than;
      enum less-than;
      enum equal;
    }
  }

  typedef bitmask-ops {
    type enumeration {
      enum match;
      enum not;
    }
  }

  typedef ip-protocol-types {
    type enumeration {
      enum icmp {
        value 1;
      }
      enum igmp {
        value 2;
      }
      enum ipip {
        value 4;
      }
      enum tcp {
        value 6;
      }
      enum egp {
        value 8;
      }
      enum udp {
        value 17;
      }
      enum pv6 {
```

```
        value 41;
      }
      enum rsvp {
        value 46;
      }
      enum gre {
        value 47;
      }
      enum esp {
        value 50;
      }
      enum ospf {
        value 89;
      }
      enum pim {
        value 103;
      }
    }
  }

  typedef dscp-remarked {
    type enumeration {
      enum be;
      enum cs1;
      enum af11;
      enum af12;
      enum af13;
      enum cs2;
      enum af21;
      enum af22;
      enum af23;
      enum cs3;
      enum af31;
      enum af32;
      enum af33;
      enum cs4;
      enum af41;
      enum af42;
      enum af43;
      enum cs5;
      enum ef;
      enum cs6;
      enum cs7;
    }
  }

  typedef dscp-type {
    type union {
```

```
      type dscp-remarked;
      type uint8;
    }
  }

  typedef fragment-mode-type {
    type enumeration {
      enum fragment {
        description
          "Indicates that fragments are checked.";
      }
      enum non-fragment {
        description
          "Indicates that non-fragmented packets are checked.";
      }
      enum fragment-spe-first {
        description
          "Indicates that the first fragmented packet is checked.";
      }
    }
  }

  /*Definition from
    http://www.ietf.org/mail-archive/web/netmod/current/msg02685.html
   */
  typedef float {
    type union {
      type decimal64 {
        fraction-digits 14;
        range "-9999.99999999999999 .. 9999.99999999999999";
      }
      type decimal64 {
        fraction-digits 13;
        range "-99999.9999999999999 .. 99999.9999999999999";
      }
      type decimal64 {
        fraction-digits 12;
        range "-999999.999999999999 .. 999999.999999999999";
      }
      type decimal64 {
        fraction-digits 11;
        range "-9999999.99999999999 .. 9999999.99999999999";
      }
      type decimal64 {
        fraction-digits 10;
        range "-99999999.9999999999 .. 99999999.9999999999";
      }
      type decimal64 {
```

```
        fraction-digits 9;
        range "-999999999.999999999 .. 999999999.999999999";
      }
      type decimal64 {
        fraction-digits 8;
        range "-9999999999.99999999 .. 9999999999.99999999";
      }
      type decimal64 {
        fraction-digits 7;
        range "-99999999999.9999999 .. 99999999999.9999999";
      }
      type decimal64 {
        fraction-digits 6;
        range "-999999999999.999999 .. 999999999999.999999";
      }
      type decimal64 {
        fraction-digits 5;
        range "-9999999999999.99999 .. 9999999999999.99999";
      }
      type decimal64 {
        fraction-digits 4;
        range "-99999999999999.9999 .. 99999999999999.9999";
      }
      type decimal64 {
        fraction-digits 3;
        range "-999999999999999.999 .. 999999999999999.999";
      }
      type decimal64 {
        fraction-digits 2;
        range "-9999999999999999.99 .. 9999999999999999.99";
      }
      type decimal64 {
        fraction-digits 1;
        range "-99999999999999999.9 .. 99999999999999999.9";
      }
    }
  }

  /* Grouping definition */
  grouping flow-filter-rule {
    container destination-prefix {
      leaf dest-prefix-length {
        type uint8;
        description
          "Specifies the mask length.";
      }
      leaf dest-prefix {
        type inet:ip-address;
```

```
        description
          "Specifies the destination address of the traffic.";
      }
    }
    container source-prefix {
      leaf src-prefix-length {
        type uint8;
        description
          "Specifies the mask length.";
      }
      leaf src-prefix {
        type inet:ip-address;
        description
          "Specifies the source IP address of the traffic.";
      }
    }
    container ip-protocol {
      list op-protocol {
        key "compare-protocol ip-proto";
        leaf compare-protocol {
          type compare-ops;
          description
            "Indicates the comparing relationship between the destination
             port number and the specified one.";
        }
        leaf ip-proto {
          type ip-protocol-types;
          description
            "Specifies a protocol type with a name or number.";
        }
      }
    }
    container port-number {
      list op-port {
        key "compare-port port";
        leaf compare-port {
          type compare-ops;
          description
            "Specifies the operation symbol of comparing source
             or destination port numbers.";
        }
        leaf port {
          type uint16;
          description
            "Specifies the port number.";
        }
      }
    }
```

```
    container destination-port {
      list op-destport {
        key "compare-destport dest-port";
        leaf compare-destport {
          type compare-ops;
          description
            "Specifies the operation symbol of comparing
             destination port numbers with designated value.";
        }
        leaf dest-port {
          type uint16;
          description
            "Specifies the destination port number of the traffic.";
        }
      }
    }
    container source-port {
      list op-srcport {
        key "compare-srcport src-port";
        leaf compare-srcport {
          type compare-ops;
          description
            "Specifies the operation symbol of comparing
             source port numbers with designated value.";
        }
        leaf src-port {
          type uint16;
          description
            "Specifies the source port number of the traffic.";
        }
      }
    }
    container icmp-type-value {
      list op-icmp-type {
        key "compare-icmp-type icmp-type";
        leaf compare-icmp-type {
          type compare-ops;
          description
            "Specifies the operation symbol of comparing
             icmp type value with designated value.";
        }
        leaf icmp-type {
          type uint8;
          description
            "Specifies the ICMP packet type.";
        }
      }
    }
```

```
    container icmp-code-value {
      list op-code-value {
        key "compare-icmp-code icmp-code";
        leaf compare-icmp-code {
          type compare-ops;
          description
            "Specifies the operation symbol of comparing
             icmp code value with designated value.";
        }
        leaf icmp-code {
          type uint8;
          description
            "Specifies the ICMP packet code.";
        }
      }
    }
    container tcp-flag {
      list op-bitmask {
        key "tcp-flag-op tcp-flag-value";
        leaf tcp-flag-op {
          type bitmask-ops;
          description
            "Indicates that traffic matching the TCP flag value
             can or can not pass the filtering rule.";
        }
        leaf tcp-flag-value {
          type uint16;
          description
            "Specifies the TCP flag value.";
        }
      }
    }
    container pkt-len {
      list op-pkt-len {
        key "compare-pkt-len packet-length";
        leaf compare-pkt-len {
          type compare-ops;
          description
            "Specifies the operation symbol of comparing
             packet length with designated value.";
        }
        leaf packet-length {
          type uint16;
          description
            "Specifies IP packet length.";
        }
      }
    }
```

```
    container dscp {
      list op-dscp {
        key "compare-dscp dscp-value";
        leaf compare-dscp {
          type compare-ops;
          description
            "Specifies the operation symbol of comparing
             dscp value with designated value.";
        }
        leaf dscp-value {
          type dscp-type;
          description
            "Specifies the DSCP value.";
        }
      }
    }
    container fragment {
      list op-fragment {
        key "compare-fragment fragment-mode";
        leaf compare-fragment {
          type bitmask-ops;
          description
            "Indicates that traffic matching the fragment type
             or not can pass the filtering rule.";
        }
        leaf fragment-mode {
          type fragment-mode-type;
          description
            "Indicates the fragment mode that is checked.";
        }
      }
    }
  }

  grouping flow-filter-action {
    container traffic-rate {
      leaf rate {
        type float;
        description
          "Specifies the traffic rate in IEEE floating point
          [IEEE.754.1985] format, units being bytes per second.";
      }
    }
    container redirect {
      leaf route-target {
        type string {
          length "3..21";
        }
```

```
        description
          "Specifies the name of a target VPN to which
           attack traffic is redirected.";
      }
    }
    container traffic-marking {
      leaf remark-dscp {
        type union {
          type dscp-remarked;
          type uint8 {
            range "0..63";
          }
        }
        description
          "Specifies a re-marked DSCP value.";
      }
    }
    container traffic-deny {
      leaf deny {
        type boolean;
        description
          "Enables a device to discard the traffic matching
           a filtering rule.";
      }
    }
  }

  grouping flow-proto-specific {
    choice protocol {
      case bgp {
        leaf peer-rid {
          type inet:ipv4-address;
        }
        leaf isrefclient {
          type boolean;
        }
        leaf med {
          type uint32;
        }
        leaf preference {
          type uint32;
        }
        leaf local-pref {
          type uint8;
        }
        leaf origin {
          type enumeration {
            enum igp {
```

```
              value "0";
            }
            enum egp {
              value "1";
            }
            enum incomplete {
              value "2";
            }
          }
        }
        leaf originator {
          type inet:ipv4-address;
        }
        leaf community {
          type string;
        }
        leaf ext-community {
          type string;
        }
        leaf cluster-list {
          type string;
        }
        leaf path-as {
          type string;
        }
      }
      case ospf {
        leaf lsa-type {
          type string;
        }
        leaf ls-id {
          type inet:ipv4-address;
        }
        leaf advertiser {
          type inet:ipv4-address;
        }
        leaf area {
          type union {
            type uint32;
            type inet:ipv4-address;
          }
        }
        leaf tag {
          type uint32;
        }
      }
      case isis {
        leaf lsp-id {
```

```
          type string;
        }
        leaf priority {
          type uint32;
        }
        leaf admin-tag {
          type uint32;
        }
      }
    }
  }

  /* Configuration data nodes */
  container flowspec-cfg {
    description
      "Configuration for flow specification.";
    list flow-route {
      key "name";
      description
        "Configuration of a flow route list.";
      leaf name {
        description
          "The name of a flow route.";
        type string;
      }
      container filter-rules {
        description
          "Configuration for filter rules.";
        uses flow-filter-rule;
      }
      container filter-actions {
        description
          "Configuration for filter actions.";
        uses flow-filter-action;
      }
    }
  }

  /* Operational state data nodes */
  container flowspec-state {
    config "false";
    description
      "Operational state of flow specification.";

    container flowspec-rib {
      list flowspec-route {
        key "index islocal";
        leaf index {
```

```
          type uint32;
          description
            "Flow Specification route index.";
        }
        leaf islocal {
          type boolean;
          description
            "Locally configured Flow Specification route.";
        }
        leaf local-name {
          type string;
          description
            "The name of locally configured Flow Specification route.";
        }
        leaf neighbor {
          type inet:ip-address;
          description
            "IP address of an advertising device";
        }
        leaf duration {
          type uint32;
          description
            "Route duration in seconds.";
        }
        container filter-rules {
          description
            "Indicate current state for filter rules.";
          uses flow-filter-rule;
        }
        container filter-actions {
          description
            "Indicate current state for filter actions.";
          uses flow-filter-action;
        }
        container protocol-specific {
          description
            "Indicate current state of protocol specific attributes.";
          uses flow-proto-specific;
        }

      }
    }
    container flowspec-statis {
    }
  }
}

//<CODE ENDS>
```

4. IANA Considerations

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registration has been made.

URI: urn:ietf:params:xml:ns:yang:ietf-flowspec

Registrant Contact: The RTGWG WG of the IETF.

XML: N/A; the requested URI is an XML namespace.

This document registers a YANG module in the "YANG Module Names" registry [RFC6020].

Name: ietf-flowspec

Namespace: urn:ietf:params:xml:ns:yang:ietf-flowspec

Prefix: flowspec

Reference: RFC XXXX

5. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

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

6. Acknowledgments

TBD.

7. References

7.1. Normative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, August 2009.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

7.2. Informative References

[I-D.liang-ospf-flowspec-extensions]
Liang, Q., You, J., and N. Wu, "OSPF Extensions for Flow Specification", draft-liang-ospf-flowspec-extensions-02 (work in progress), November 2014.

[I-D.you-isis-flowspec-extensions]
You, J. and Q. Liang, "IS-IS Extensions for Flow Specification", draft-you-isis-flowspec-extensions-00 (work in progress), September 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

Authors' Addresses

Nan Wu
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: eric.wu@huawei.com

Shunwan Zhuang
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com