

IDR Working Group  
Internet-Draft  
Obsoletes: [5575](#),7674 (if approved)  
Intended status: Standards Track  
Expires: April 18, 2021

C. Loibl  
next layer Telekom GmbH  
S. Hares  
Huawei  
R. Raszuk  
Bloomberg LP  
D. McPherson  
Verisign  
M. Bacher  
T-Mobile Austria  
October 15, 2020

**Dissemination of Flow Specification Rules**  
**draft-ietf-idr-rfc5575bis-27**

Abstract

This document defines a Border Gateway Protocol Network Layer Reachability Information (BGP NLRI) encoding format that can be used to distribute traffic Flow Specifications. This allows the routing system to propagate information regarding more specific components of the traffic aggregate defined by an IP destination prefix.

It also specifies BGP Extended Community encoding formats, that can be used to propagate Traffic Filtering Actions along with the Flow Specification NLRI. Those Traffic Filtering Actions encode actions a routing system can take if the packet matches the Flow Specification.

Additionally, it defines two applications of that encoding format: one that can be used to automate inter-domain coordination of traffic filtering, such as what is required in order to mitigate (distributed) denial-of-service attacks, and a second application to provide traffic filtering in the context of a BGP/MPLS VPN service. Other applications (e.g. centralized control of traffic in a SDN or NFV context) are also possible. Other documents may specify Flow Specification extensions.

The information is carried via BGP, thereby reusing protocol algorithms, operational experience, and administrative processes such as inter-provider peering agreements.

This document obsoletes both [RFC5575](#) and [RFC7674](#).

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 2021.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Definitions of Terms Used in This Memo . . . . .	<a href="#">5</a>
<a href="#">3.</a>	Flow Specifications . . . . .	<a href="#">5</a>
<a href="#">4.</a>	Dissemination of IPv4 Flow Specification Information . . . . .	<a href="#">6</a>
<a href="#">4.1.</a>	Length Encoding . . . . .	<a href="#">7</a>
<a href="#">4.2.</a>	NLRI Value Encoding . . . . .	<a href="#">7</a>
<a href="#">4.2.1.</a>	Operators . . . . .	<a href="#">7</a>
<a href="#">4.2.2.</a>	Components . . . . .	<a href="#">9</a>
<a href="#">4.3.</a>	Examples of Encodings . . . . .	<a href="#">14</a>
<a href="#">5.</a>	Traffic Filtering . . . . .	<a href="#">16</a>
<a href="#">5.1.</a>	Ordering of Flow Specifications . . . . .	<a href="#">17</a>
<a href="#">6.</a>	Validation Procedure . . . . .	<a href="#">18</a>
<a href="#">7.</a>	Traffic Filtering Actions . . . . .	<a href="#">19</a>
7.1.	Traffic Rate in Bytes (traffic-rate-bytes) sub-type 0x06	21



7.2. Traffic Rate in Packets (traffic-rate-packets) sub-type	
TBD . . . . .	<a href="#">21</a>
<a href="#">7.3.</a> Traffic-action (traffic-action) sub-type 0x07 . . . . .	<a href="#">21</a>
<a href="#">7.4.</a> RT Redirect (rt-redirect) sub-type 0x08 . . . . .	<a href="#">22</a>
<a href="#">7.5.</a> Traffic Marking (traffic-marking) sub-type 0x09 . . . . .	<a href="#">23</a>
7.6. Interaction with other Filtering Mechanisms in Routers .	23
7.7. Considerations on Traffic Filtering Action Interference .	24
8. Dissemination of Traffic Filtering in BGP/MPLS VPN Networks .	24
<a href="#">9.</a> Traffic Monitoring . . . . .	<a href="#">25</a>
<a href="#">10.</a> Error Handling . . . . .	<a href="#">25</a>
<a href="#">11.</a> IANA Considerations . . . . .	<a href="#">25</a>
<a href="#">11.1.</a> AFI/SAFI Definitions . . . . .	<a href="#">25</a>
<a href="#">11.2.</a> Flow Component Definitions . . . . .	<a href="#">27</a>
<a href="#">11.3.</a> Extended Community Flow Specification Actions . . . . .	<a href="#">28</a>
<a href="#">12.</a> Security Considerations . . . . .	<a href="#">30</a>
<a href="#">13.</a> Contributors . . . . .	<a href="#">32</a>
<a href="#">14.</a> Acknowledgements . . . . .	<a href="#">32</a>
<a href="#">15.</a> References . . . . .	<a href="#">32</a>
<a href="#">15.1.</a> Normative References . . . . .	<a href="#">32</a>
<a href="#">15.2.</a> Informative References . . . . .	<a href="#">34</a>
<a href="#">15.3.</a> URIs . . . . .	<a href="#">35</a>
<a href="#">Appendix A.</a> Example Python code: flow_rule_cmp . . . . .	<a href="#">35</a>
<a href="#">Appendix B.</a> Comparison with <a href="#">RFC 5575</a> . . . . .	<a href="#">38</a>
Authors' Addresses . . . . .	<a href="#">39</a>

## [1.](#) Introduction

This document obsoletes "Dissemination of Flow Specification Rules" [[RFC5575](#)] (see [Appendix B](#) for the differences). This document also obsoletes "Clarification of the Flowspec Redirect Extended Community" [[RFC7674](#)] since it incorporates the encoding of the BGP Flow Specification Redirect Extended Community in [Section 7.4](#).

Modern IP routers have the capability to forward traffic and to classify, shape, rate limit, filter, or redirect packets based on administratively defined policies. These traffic policy mechanisms allow the operator to define match rules that operate on multiple fields of the packet header. Actions such as the ones described above can be associated with each rule.

The n-tuple consisting of the matching criteria defines an aggregate traffic Flow Specification. The matching criteria can include elements such as source and destination address prefixes, IP protocol, and transport protocol port numbers.

[Section 4](#) of this document defines a general procedure to encode Flow Specifications for aggregated traffic flows so that they can be distributed as a BGP [[RFC4271](#)] NLRI. Additionally, [Section 7](#) of this



document defines the required Traffic Filtering Actions BGP Extended Communities and mechanisms to use BGP for intra- and inter-provider distribution of traffic filtering rules to filter (distributed) denial-of-service (DoS) attacks.

By expanding routing information with Flow Specifications, the routing system can take advantage of the ACL (Access Control List) or firewall capabilities in the router's forwarding path. Flow Specifications can be seen as more specific routing entries to a unicast prefix and are expected to depend upon the existing unicast data information.

A Flow Specification received from an external autonomous system will need to be validated against unicast routing before being accepted ([Section 6](#)). The Flow Specification received from an internal BGP peer within the same autonomous system [[RFC4271](#)] is assumed to have been validated prior to transmission within the internal BGP (iBGP) mesh of an autonomous system. If the aggregate traffic flow defined by the unicast destination prefix is forwarded to a given BGP peer, then the local system can install more specific Flow Specifications that may result in different forwarding behavior, as requested by this system.

From an operational perspective, the utilization of BGP as the carrier for this information allows a network service provider to reuse both internal route distribution infrastructure (e.g., route reflector or confederation design) and existing external relationships (e.g., inter-domain BGP sessions to a customer network).

While it is certainly possible to address this problem using other mechanisms, this solution has been utilized in deployments because of the substantial advantage of being an incremental addition to already deployed mechanisms.

In current deployments, the information distributed by this extension is originated both manually as well as automatically, the latter by systems that are able to detect malicious traffic flows. When automated systems are used, care should be taken to ensure the correctness of the automated system. The the limitations of the receiving systems that need to process these automated Flow Specifications need to be taken in consideration as well (see also [Section 12](#)).

This specification defines required protocol extensions to address most common applications of IPv4 unicast and VPNv4 unicast filtering. The same mechanism can be reused and new match criteria added to



address similar filtering needs for other BGP address families such as IPv6 families [[I-D.ietf-idr-flow-spec-v6](#)].

## 2. Definitions of Terms Used in This Memo

AFI - Address Family Identifier.

AS - Autonomous System.

Loc-RIB - The Loc-RIB contains the routes that have been selected by the local BGP speaker's Decision Process [[RFC4271](#)].

NLRI - Network Layer Reachability Information.

PE - Provider Edge router.

RIB - Routing Information Base.

SAFI - Subsequent Address Family Identifier.

VRF - Virtual Routing and Forwarding instance.

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

## 3. Flow Specifications

A Flow Specification is an n-tuple consisting of several matching criteria that can be applied to IP traffic. A given IP packet is said to match the defined Flow Specification if it matches all the specified criteria. This n-tuple is encoded into a BGP NLRI defined below.

A given Flow Specification may be associated with a set of attributes, depending on the particular application; such attributes may or may not include reachability information (i.e., NEXT\_HOP). Well-known or AS-specific community attributes can be used to encode a set of predetermined actions.

A particular application is identified by a specific (Address Family Identifier, Subsequent Address Family Identifier (AFI, SAFI)) pair [[RFC4760](#)] and corresponds to a distinct set of RIBs. Those RIBs should be treated independently from each other in order to assure non-interference between distinct applications.





BGP itself treats the NLRI as a key to an entry in its databases. Entries that are placed in the Loc-RIB are then associated with a given set of semantics, which is application dependent. This is consistent with existing BGP applications. For instance, IP unicast routing (AFI=1, SAFI=1) and IP multicast reverse-path information (AFI=1, SAFI=2) are handled by BGP without any particular semantics being associated with them until installed in the Loc-RIB.

Standard BGP policy mechanisms, such as UPDATE filtering by NLRI prefix as well as community matching and must apply to the Flow specification defined NLRI-type. Network operators can also control propagation of such routing updates by enabling or disabling the exchange of a particular (AFI, SAFI) pair on a given BGP peering session.

#### 4. Dissemination of IPv4 Flow Specification Information

This document defines a Flow Specification NLRI type (Figure 1) that may include several components such as destination prefix, source prefix, protocol, ports, and others (see [Section 4.2](#) below).

This NLRI information is encoded using MP\_REACH\_NLRI and MP\_UNREACH\_NLRI attributes as defined in [\[RFC4760\]](#). When advertising Flow Specifications, the Length of Next Hop Network Address MUST be set to 0. The Network Address of Next Hop field MUST be ignored.

The NLRI field of the MP\_REACH\_NLRI and MP\_UNREACH\_NLRI is encoded as one or more 2-tuples of the form <length, NLRI value>. It consists of a 1- or 2-octet length field followed by a variable-length NLRI value. The length is expressed in octets.

```

+-----+
| length (0xnn or 0xfnnn) |
+-----+
| NLRI value (variable) |
+-----+

```

Figure 1: Flow Specification NLRI for IPv4

Implementations wishing to exchange Flow Specification MUST use BGP's Capability Advertisement facility to exchange the Multiprotocol Extension Capability Code (Code 1) as defined in [\[RFC4760\]](#). The (AFI, SAFI) pair carried in the Multiprotocol Extension Capability MUST be (AFI=1, SAFI=133) for IPv4 Flow Specification, and (AFI=1, SAFI=134) for VPNv4 Flow Specification.



#### **4.1. Length Encoding**

- o If the NLRI length is smaller than 240 (0xf0 hex) octets, the length field can be encoded as a single octet.
- o Otherwise, it is encoded as an extended-length 2-octet value in which the most significant nibble has the hex value 0xf.

In Figure 1 above, values less-than 240 are encoded using two hex digits (0xnn). Values above 239 are encoded using 3 hex digits (0xfnnn). The highest value that can be represented with this encoding is 4095. For example the length value of 239 is encoded as 0xef (single octet) while 240 is encoded as 0xf0f0 (2-octet).

#### **4.2. NLRI Value Encoding**

The Flow Specification NLRI value consists of a list of optional components and is encoded as follows:

Encoding: <[component]+>

A specific packet is considered to match the Flow Specification when it matches the intersection (AND) of all the components present in the Flow Specification.

Components MUST follow strict type ordering by increasing numerical order. A given component type MAY (exactly once) be present in the Flow Specification. If present, it MUST precede any component of higher numeric type value.

All combinations of components within a single Flow Specification are allowed. However, some combinations cannot match any packets (e.g. "ICMP Type AND Port" will never match any packets), and thus SHOULD NOT be propagated by BGP.

A NLRI value not encoded as specified here, including a NLRI that contains an unknown component type, is considered malformed and error handling according to [Section 10](#) is performed.

##### **4.2.1. Operators**

Most of the components described below make use of comparison operators. Which of the two operators is used is defined by the components in [Section 4.2.2](#). The operators are encoded as a single octet.



#### 4.2.1.1. Numeric Operator (numeric\_op)

This operator is encoded as shown in Figure 2.

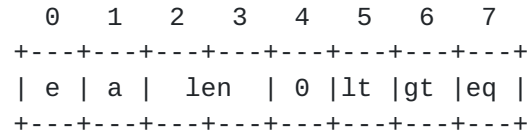


Figure 2: Numeric Operator (numeric\_op)

- e - end-of-list bit: Set in the last {op, value} pair in the list.
- a - AND bit: If unset, the result of the previous {op, value} pair is logically ORed with the current one. If set, the operation is a logical AND. In the first operator octet of a sequence it MUST be encoded as unset and MUST be treated as always unset on decoding. The AND operator has higher priority than OR for the purposes of evaluating logical expressions.
- len - length: The length of the value field for this operator given as  $(1 \ll \text{len})$ . This encodes 1 (len=00), 2 (len=01), 4 (len=10), 8 (len=11) octets.
- 0 - MUST be set to 0 on NLRI encoding, and MUST be ignored during decoding
- lt - less than comparison between data and value.
- gt - greater than comparison between data and value.
- eq - equality between data and value.

The bits lt, gt, and eq can be combined to produce common relational operators such as "less or equal", "greater or equal", and "not equal to" as shown in Table 1.



lt	gt	eq	Resulting operation
0	0	0	false (independent of the value)
0	0	1	== (equal)
0	1	0	> (greater than)
0	1	1	>= (greater than or equal)
1	0	0	< (less than)
1	0	1	<= (less than or equal)
1	1	0	!= (not equal value)
1	1	1	true (independent of the value)

Table 1: Comparison operation combinations

#### 4.2.1.2. Bitmask Operator (bitmask\_op)

This operator is encoded as shown in Figure 3.

0	1	2	3	4	5	6	7
e	a	len	0	0	not	m	

Figure 3: Bitmask Operator (bitmask\_op)

e, a, len - Most significant nibble: (end-of-list bit, AND bit, and length field), as defined in the Numeric Operator format in [Section 4.2.1.1](#).

not - NOT bit: If set, logical negation of operation.

m - Match bit: If set, this is a bitwise match operation defined as "(data AND value) == value"; if unset, (data AND value) evaluates to TRUE if any of the bits in the value mask are set in the data

0 - all 0 bits: MUST be set to 0 on NLRI encoding, and MUST be ignored during decoding

#### 4.2.2. Components

The encoding of each of the components begins with a type field (1 octet) followed by a variable length parameter. The following sections define component types and parameter encodings for the IPv4 IP layer and transport layer headers. IPv6 NLRI component types are described in [\[I-D.ietf-idr-flow-spec-v6\]](#).





#### **4.2.2.1. Type 1 - Destination Prefix**

Encoding: <type (1 octet), length (1 octet), prefix (variable)>

Defines the destination prefix to match. The length and prefix fields are encoded as in BGP UPDATE messages [[RFC4271](#)]

#### **4.2.2.2. Type 2 - Source Prefix**

Encoding: <type (1 octet), length (1 octet), prefix (variable)>

Defines the source prefix to match. The length and prefix fields are encoded as in BGP UPDATE messages [[RFC4271](#)]

#### **4.2.2.3. Type 3 - IP Protocol**

Encoding: <type (1 octet), [numeric\_op, value]+>

Contains a list of {numeric\_op, value} pairs that are used to match the IP protocol value octet in IP packet header (see [[RFC0791](#) [Section 3.1](#)]).

This component uses the Numeric Operator (numeric\_op) described in [Section 4.2.1.1](#). Type 3 component values SHOULD be encoded as single octet (numeric\_op len=00).

#### **4.2.2.4. Type 4 - Port**

Encoding: <type (1 octet), [numeric\_op, value]+>

Defines a list of {numeric\_op, value} pairs that matches source OR destination TCP/UDP ports (see [[RFC0793](#) [Section 3.1](#)] and [[RFC0768](#) [Section "Format"](#)]). This component matches if either the destination port OR the source port of a IP packet matches the value.

This component uses the Numeric Operator (numeric\_op) described in [Section 4.2.1.1](#). Type 4 component values SHOULD be encoded as 1- or 2-octet quantities (numeric\_op len=00 or len=01).

In case of the presence of the port (destination-port [Section 4.2.2.5](#), source-port [Section 4.2.2.6](#)) component only TCP or UDP packets can match the entire Flow Specification. The port component, if present, never matches when the packet's IP protocol value is not 6 (TCP) or 17 (UDP), if the packet is fragmented and this is not the first fragment, or if the system is unable to locate the transport header. Different implementations may or may not be able to decode the transport header in the presence of IP options or Encapsulating Security Payload (ESP) NULL [[RFC4303](#)] encryption.



#### **4.2.2.5. Type 5 - Destination Port**

Encoding: <type (1 octet), [numeric\_op, value]+>

Defines a list of {numeric\_op, value} pairs used to match the destination port of a TCP or UDP packet (see also [\[RFC0793\] Section 3.1](#) and [\[RFC0768\] Section "Format"](#)).

This component uses the Numeric Operator (numeric\_op) described in [Section 4.2.1.1](#). Type 5 component values SHOULD be encoded as 1- or 2-octet quantities (numeric\_op len=00 or len=01).

The last paragraph of [Section 4.2.2.4](#) also applies to this component.

#### **4.2.2.6. Type 6 - Source Port**

Encoding: <type (1 octet), [numeric\_op, value]+>

Defines a list of {numeric\_op, value} pairs used to match the source port of a TCP or UDP packet (see also [\[RFC0793\] Section 3.1](#) and [\[RFC0768\] Section "Format"](#)).

This component uses the Numeric Operator (numeric\_op) described in [Section 4.2.1.1](#). Type 6 component values SHOULD be encoded as 1- or 2-octet quantities (numeric\_op len=00 or len=01).

The last paragraph of [Section 4.2.2.4](#) also applies to this component.

#### **4.2.2.7. Type 7 - ICMP type**

Encoding: <type (1 octet), [numeric\_op, value]+>

Defines a list of {numeric\_op, value} pairs used to match the type field of an ICMP packet (see also [\[RFC0792\] Section "Message Formats"](#)).

This component uses the Numeric Operator (numeric\_op) described in [Section 4.2.1.1](#). Type 7 component values SHOULD be encoded as single octet (numeric\_op len=00).

In case of the presence of the ICMP type component only ICMP packets can match the entire Flow Specification. The ICMP type component, if present, never matches when the packet's IP protocol value is not 1 (ICMP), if the packet is fragmented and this is not the first fragment, or if the system is unable to locate the transport header. Different implementations may or may not be able to decode the transport header in the presence of IP options or Encapsulating Security Payload (ESP) NULL [\[RFC4303\]](#) encryption.



#### **4.2.2.8. Type 8 - ICMP code**

Encoding: <type (1 octet), [numeric\_op, value]+>

Defines a list of {numeric\_op, value} pairs used to match the code field of an ICMP packet (see also [[RFC0792](#)] Section "Message Formats").

This component uses the Numeric Operator (numeric\_op) described in [Section 4.2.1.1](#). Type 8 component values SHOULD be encoded as single octet (numeric\_op len=00).

In case of the presence of the ICMP code component only ICMP packets can match the entire Flow Specification. The ICMP code component, if present, never matches when the packet's IP protocol value is not 1 (ICMP), if the packet is fragmented and this is not the first fragment, or if the system is unable to locate the transport header. Different implementations may or may not be able to decode the transport header in the presence of IP options or Encapsulating Security Payload (ESP) NULL [[RFC4303](#)] encryption.

#### **4.2.2.9. Type 9 - TCP flags**

Encoding: <type (1 octet), [bitmask\_op, bitmask]+>

Defines a list of {bitmask\_op, bitmask} pairs used to match TCP Control Bits (see also [[RFC0793](#)] [Section 3.1](#)).

This component uses the Bitmask Operator (bitmask\_op) described in [Section 4.2.1.2](#). Type 9 component bitmasks MUST be encoded as 1- or 2-octet bitmask (bitmask\_op len=00 or len=01).

When a single octet (bitmask\_op len=00) is specified, it matches octet 14 of the TCP header (see also [[RFC0793](#)] [Section 3.1](#)), which contains the TCP Control Bits. When a 2-octet (bitmask\_op len=01) encoding is used, it matches octets 13 and 14 of the TCP header with the data offset (leftmost 4 bits) always treated as 0.

In case of the presence of the TCP flags component only TCP packets can match the entire Flow Specification. The TCP flags component, if present, never matches when the packet's IP protocol value is not 6 (TCP), if the packet is fragmented and this is not the first fragment, or if the system is unable to locate the transport header. Different implementations may or may not be able to decode the transport header in the presence of IP options or Encapsulating Security Payload (ESP) NULL [[RFC4303](#)] encryption.



**4.2.2.10. Type 10 - Packet length**

Encoding: <type (1 octet), [numeric\_op, value]+>

Defines a list of {numeric\_op, value} pairs used to match on the total IP packet length (excluding Layer 2 but including IP header).

This component uses the Numeric Operator (numeric\_op) described in [Section 4.2.1.1](#). Type 10 component values SHOULD be encoded as 1- or 2-octet quantities (numeric\_op len=00 or len=01).

**4.2.2.11. Type 11 - DSCP (Diffserv Code Point)**

Encoding: <type (1 octet), [numeric\_op, value]+>

Defines a list of {numeric\_op, value} pairs used to match the 6-bit DSCP field (see also [RFC2474](#)).

This component uses the Numeric Operator (numeric\_op) described in [Section 4.2.1.1](#). Type 11 component values MUST be encoded as single octet (numeric\_op len=00).

The six least significant bits contain the DSCP value. All other bits SHOULD be treated as 0.

**4.2.2.12. Type 12 - Fragment**

Encoding: <type (1 octet), [bitmask\_op, bitmask]+>

Defines a list of {bitmask\_op, bitmask} pairs used to match specific IP fragments.

This component uses the Bitmask Operator (bitmask\_op) described in [Section 4.2.1.2](#). The Type 12 component bitmask MUST be encoded as single octet bitmask (bitmask\_op len=00).

```

      0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+
| 0 | 0 | 0 | 0 | LF | FF | IsF | DF |
+---+---+---+---+---+---+---+

```

Figure 4: Fragment Bitmask Operand

Bitmask values:

DF - Don't fragment - match if [RFC0791](#) IP Header Flags Bit-1 (DF) is 1





- IsF - Is a fragment other than the first - match if [[RFC0791](#)] IP Header Fragment Offset is not 0
- FF - First fragment - match if [[RFC0791](#)] IP Header Fragment Offset is 0 AND Flags Bit-2 (MF) is 1
- LF - Last fragment - match if [[RFC0791](#)] IP Header Fragment Offset is not 0 AND Flags Bit-2 (MF) is 0
- 0 - MUST be set to 0 on NLRI encoding, and MUST be ignored during decoding

### [4.3.](#) Examples of Encodings

#### [4.3.1.](#) Example 1

An example of a Flow Specification NLRI encoding for: "all packets to 192.0.2.0/24 and TCP port 25".

```
+-----+-----+-----+-----+
| length | destination   | protocol | port   |
+-----+-----+-----+-----+
| 0x0b   | 01 18 c0 00 02 | 03 81 06 | 04 81 19 |
+-----+-----+-----+-----+
```

Decoded:

```
+-----+-----+-----+-----+
| Value |           |
+-----+-----+-----+-----+
| 0x0b | length   | 11 octets (len<240 1-octet) |
| 0x01 | type     | Type 1 - Destination Prefix |
| 0x18 | length   | 24 bit                       |
| 0xc0 | prefix   | 192                           |
| 0x00 | prefix   | 0                             |
| 0x02 | prefix   | 2                             |
| 0x03 | type     | Type 3 - IP Protocol         |
| 0x81 | numeric_op | end-of-list, value size=1, == |
| 0x06 | value     | 6 (TCP)                      |
| 0x04 | type     | Type 4 - Port                |
| 0x81 | numeric_op | end-of-list, value size=1, == |
| 0x19 | value     | 25                           |
+-----+-----+-----+-----+
```

This constitutes a NLRI with a NLRI length of 11 octets.



#### 4.3.2. Example 2

An example of a Flow Specification NLRI encoding for: "all packets to 192.0.2.0/24 from 203.0.113.0/24 and port {range [137, 139] or 8080}".

```
+-----+-----+-----+-----+
| length | destination | source | port |
+-----+-----+-----+-----+
| 0x12 | 01 18 c0 00 02 | 02 18 cb 00 71 | 04 03 89 45 8b 91 1f 90 |
+-----+-----+-----+-----+
```

Decoded:

```
+-----+-----+-----+
| Value | | |
+-----+-----+-----+
| 0x12 | length | 18 octets (len<240 1-octet) |
| 0x01 | type | Type 1 - Destination Prefix |
| 0x18 | length | 24 bit |
| 0xc0 | prefix | 192 |
| 0x00 | prefix | 0 |
| 0x02 | prefix | 2 |
| 0x02 | type | Type 2 - Source Prefix |
| 0x18 | length | 24 bit |
| 0xcb | prefix | 203 |
| 0x00 | prefix | 0 |
| 0x71 | prefix | 113 |
| 0x04 | type | Type 4 - Port |
| 0x03 | numeric_op | value size=1, >= |
| 0x89 | value | 137 |
| 0x45 | numeric_op | "AND", value size=1, <= |
| 0x8b | value | 139 |
| 0x91 | numeric_op | end-of-list, value size=2, == |
| 0x1f90 | value | 8080 |
+-----+-----+-----+
```

This constitutes a NLRI with a NLRI length of 18 octets.

#### 4.3.3. Example 3

An example of a Flow Specification NLRI encoding for: "all packets to 192.0.2.1/32 and fragment { DF or FF } (matching packet with DF bit set or First Fragments)



```

+-----+-----+-----+
| length | destination      | fragment |
+-----+-----+-----+
| 0x09  | 01 20 c0 00 02 01 | 0c 80 05 |
+-----+-----+-----+

```

Decoded:

```

+-----+-----+-----+
| Value |           |           |
+-----+-----+-----+
| 0x09 | length    | 9 octets (len<240 1-octet) |
| 0x01 | type      | Type 1 - Destination Prefix |
| 0x20 | length    | 32 bit |
| 0xc0 | prefix    | 192 |
| 0x00 | prefix    | 0 |
| 0x02 | prefix    | 2 |
| 0x01 | prefix    | 1 |
| 0x0c | type      | Type 12 - Fragment |
| 0x80 | bitmask_op | end-of-list, value size=1 |
| 0x05 | bitmask    | DF=1, FF=1 |
+-----+-----+-----+

```

This constitutes a NLRI with a NLRI length of 9 octets.

## 5. Traffic Filtering

Traffic filtering policies have been traditionally considered to be relatively static. Limitations of these static mechanisms caused this new dynamic mechanism to be designed for the three new applications of traffic filtering:

- o Prevention of traffic-based, denial-of-service (DOS) attacks.
- o Traffic filtering in the context of BGP/MPLS VPN service.
- o Centralized traffic control for SDN/NFV networks.

These applications require coordination among service providers and/or coordination among the AS within a service provider.

The Flow Specification NLRI defined in [Section 4](#) conveys information about traffic filtering rules for traffic that should be discarded or handled in a manner specified by a set of pre-defined actions (which are defined in BGP Extended Communities). This mechanism is primarily designed to allow an upstream autonomous system to perform inbound filtering in their ingress routers of traffic that a given downstream AS wishes to drop.



In order to achieve this goal, this document specifies two application-specific NLRI identifiers that provide traffic filters, and a set of actions encoding in BGP Extended Communities. The two application-specific NLRI identifiers are:

- o IPv4 Flow Specification identifier (AFI=1, SAFI=133) along with specific semantic rules for IPv4 routes, and
- o VPNv4 Flow Specification identifier (AFI=1, SAFI=134) value, which can be used to propagate traffic filtering information in a BGP/MPLS VPN environment.

Encoding of the NLRI is described in [Section 4](#) for IPv4 Flow Specification and in [Section 8](#) for VPNv4 Flow Specification. The filtering actions are described in [Section 7](#).

### **5.1. Ordering of Flow Specifications**

More than one Flow Specification may match a particular traffic flow. Thus, it is necessary to define the order in which Flow Specifications get matched and actions being applied to a particular traffic flow. This ordering function is such that it does not depend on the arrival order of the Flow Specification via BGP and thus is consistent in the network.

The relative order of two Flow Specifications is determined by comparing their respective components. The algorithm starts by comparing the left-most components (lowest component type value) of the Flow Specifications. If the types differ, the Flow Specification with lowest numeric type value has higher precedence (and thus will match before) than the Flow Specification that doesn't contain that component type. If the component types are the same, then a type-specific comparison is performed (see below). If the types are equal the algorithm continues with the next component.

For IP prefix values (IP destination or source prefix): If one of the two prefixes to compare is a more specific prefix of the other, the more specific prefix has higher precedence. Otherwise the one with the lowest IP value has higher precedence.

For all other component types, unless otherwise specified, the comparison is performed by comparing the component data as a binary string using the memcmp() function as defined by [\[ISO IEC 9899\]](#). For strings with equal lengths the lowest string (memcmp) has higher precedence. For strings of different lengths, the common prefix is compared. If the common prefix is not equal the string with the lowest prefix has higher precedence. If the common prefix is equal,





the longest string is considered to have higher precedence than the shorter one.

The code in [Appendix A](#) shows a Python3 implementation of the comparison algorithm. The full code was tested with Python 3.6.3 and can be obtained at <https://github.com/stoffi92/rfc5575bis/tree/master/flowspec-cmp> [1].

## 6. Validation Procedure

Flow Specifications received from a BGP peer that are accepted in the respective Adj-RIB-In are used as input to the route selection process. Although the forwarding attributes of two routes for the same Flow Specification prefix may be the same, BGP is still required to perform its path selection algorithm in order to select the correct set of attributes to advertise.

The first step of the BGP Route Selection procedure ([Section 9.1.2 of \[RFC4271\]](#)) is to exclude from the selection procedure routes that are considered non-feasible. In the context of IP routing information, this step is used to validate that the NEXT\_HOP attribute of a given route is resolvable.

The concept can be extended, in the case of the Flow Specification NLRI, to allow other validation procedures.

The validation process described below validates Flow Specifications against unicast routes received over the same AFI but the associated unicast routing information SAFI:

Flow Specification received over SAFI=133 will be validated against routes received over SAFI=1

Flow Specification received over SAFI=134 will be validated against routes received over SAFI=128

In the absence of explicit configuration a Flow Specification NLRI MUST be validated such that it is considered feasible if and only if all of the conditions below are true:

- a) A destination prefix component is embedded in the Flow Specification.
- b) The originator of the Flow Specification matches the originator of the best-match unicast route for the destination prefix embedded in the Flow Specification (this is the unicast route with the longest possible prefix length covering the destination prefix embedded in the Flow Specification).



c) There are no "more-specific" unicast routes, when compared with the flow destination prefix, that have been received from a different neighboring AS than the best-match unicast route, which has been determined in rule b).

However, rule a) MAY be relaxed by explicit configuration, permitting Flow Specifications that include no destination prefix component. If such is the case, rules b) and c) are moot and MUST be disregarded.

By "originator" of a BGP route, we mean either the address of the originator in the ORIGINATOR\_ID Attribute [[RFC4456](#)], or the source IP address of the BGP peer, if this path attribute is not present.

BGP implementations MUST also enforce that the AS\_PATH attribute of a route received via the External Border Gateway Protocol (eBGP) contains the neighboring AS in the left-most position of the AS\_PATH attribute. While this rule is optional in the BGP specification, it becomes necessary to enforce it here for security reasons.

The best-match unicast route may change over the time independently of the Flow Specification NLRI. Therefore, a revalidation of the Flow Specification NLRI MUST be performed whenever unicast routes change. Revalidation is defined as retesting rules a) to c) as described above.

Explanation:

The underlying concept is that the neighboring AS that advertises the best unicast route for a destination is allowed to advertise Flow Specification information that conveys a destination prefix that is more or equally specific. Thus, as long as there are no "more-specific" unicast routes, received from a different neighboring AS, which would be affected by that Flow Specification, the Flow Specification is validated successfully.

The neighboring AS is the immediate destination of the traffic described by the Flow Specification. If it requests these flows to be dropped, that request can be honored without concern that it represents a denial of service in itself. The reasoning is that this is as if the traffic is being dropped by the downstream autonomous system, and there is no added value in carrying the traffic to it.

## **7. Traffic Filtering Actions**

This document defines a minimum set of Traffic Filtering Actions that it standardizes as BGP extended communities [[RFC4360](#)]. This is not meant to be an inclusive list of all the possible actions, but only a subset that can be interpreted consistently across the network.



Additional actions can be defined as either requiring standards or as vendor specific.

The default action for a matching Flow Specification is to accept the packet (treat the packet according to the normal forwarding behaviour of the system).

This document defines the following extended communities values shown in Table 2 in the form 0xttss where tt indicates the type and ss indicates the sub-type of the extended community. Encodings for these extended communities are described below.

community 0xttss	action	encoding
0x8006	traffic-rate-bytes ( <a href="#">Section 7.1</a> )	2-octet AS, 4-octet float
TBD	traffic-rate-packets ( <a href="#">Section 7.1</a> )	2-octet AS, 4-octet float
0x8007	traffic-action ( <a href="#">Section 7.3</a> )	bitmask
0x8008	rt-redirect AS-2octet ( <a href="#">Section 7.4</a> )	2-octet AS, 4-octet value
0x8108	rt-redirect IPv4 ( <a href="#">Section 7.4</a> )	4-octet IPv4 address, 2-octet value
0x8208	rt-redirect AS-4octet ( <a href="#">Section 7.4</a> )	4-octet AS, 2-octet value
0x8009	traffic-marking ( <a href="#">Section 7.5</a> )	DSCP value

Table 2: Traffic Filtering Action Extended Communities

Multiple Traffic Filtering Actions defined in this document may be present for a single Flow Specification and SHOULD be applied to the traffic flow (for example traffic-rate-bytes and rt-redirect can be applied to packets at the same time). If not all of the Traffic Filtering Actions can be applied to a traffic flow they should be treated as interfering Traffic Filtering Actions (see below).

Some Traffic Filtering Actions may interfere with each other or even contradict. [Section 7.7](#) of this document provides general considerations on such Traffic Filtering Action interference. Any additional definition of Traffic Filtering Actions SHOULD specify the action to take if those Traffic Filtering Actions interfere (also with existing Traffic Filtering Actions).



All Traffic Filtering Actions are specified as transitive BGP Extended Communities.

### **7.1. Traffic Rate in Bytes (traffic-rate-bytes) sub-type 0x06**

The traffic-rate-bytes extended community uses the following extended community encoding:

The first two octets carry the 2-octet id, which can be assigned from a 2-octet AS number. When a 4-octet AS number is locally present, the 2 least significant octets of such an AS number can be used. This value is purely informational and SHOULD NOT be interpreted by the implementation.

The remaining 4 octets carry the maximum rate information in IEEE floating point [[IEEE.754.1985](#)] format, units being bytes per second. A traffic-rate of 0 should result on all traffic for the particular flow to be discarded. On encoding the traffic-rate MUST NOT be negative. On decoding negative values MUST be treated as zero (discard all traffic).

Interferes with: May interfere with the traffic-rate-packets (see [Section 7.2](#)). A policy may allow both filtering by traffic-rate-packets and traffic-rate-bytes. If the policy does not allow this, these two actions will conflict.

### **7.2. Traffic Rate in Packets (traffic-rate-packets) sub-type TBD**

The traffic-rate-packets extended community uses the same encoding as the traffic-rate-bytes extended community. The floating point value carries the maximum packet rate in packets per second. A traffic-rate-packets of 0 should result in all traffic for the particular flow to be discarded. On encoding the traffic-rate-packets MUST NOT be negative. On decoding negative values MUST be treated as zero (discard all traffic).

Interferes with: May interfere with the traffic-rate-bytes (see [Section 7.1](#)). A policy may allow both filtering by traffic-rate-packets and traffic-rate-bytes. If the policy does not allow this, these two actions will conflict.

### **7.3. Traffic-action (traffic-action) sub-type 0x07**

The traffic-action extended community consists of 6 octets of which only the 2 least significant bits of the 6th octet (from left to right) are defined by this document as shown in Figure 5.





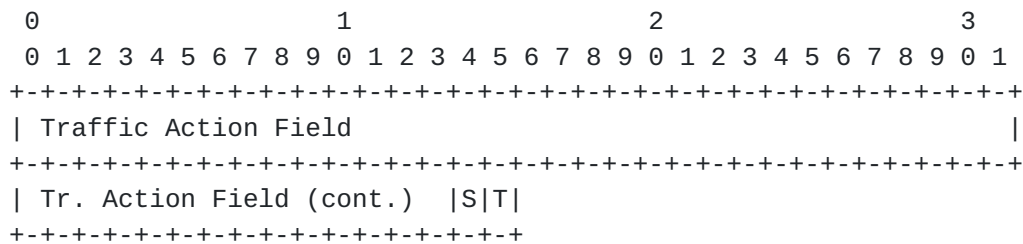


Figure 5: Traffic-action Extended Community Encoding

where S and T are defined as:

- o T: Terminal Action (bit 47): When this bit is set, the traffic filtering engine will evaluate any subsequent Flow Specifications (as defined by the ordering procedure [Section 5.1](#)). If not set, the evaluation of the traffic filters stops when this Flow Specification is evaluated.
- o S: Sample (bit 46): Enables traffic sampling and logging for this Flow Specification (only effective when set).
- o Traffic Action Field: Other Traffic Action Field (see [Section 11](#)) bits unused in this specification. These bits MUST be set to 0 on encoding, and MUST be ignored during decoding.

The use of the Terminal Action (bit 47) may result in more than one Flow Specification matching a particular traffic flow. All the Traffic Filtering Actions from these Flow Specifications shall be collected and applied. In case of interfering Traffic Filtering Actions it is an implementation decision which Traffic Filtering Actions are selected. See also [Section 7.7](#).

Interferes with: No other BGP Flow Specification Traffic Filtering Action in this document.

#### [7.4](#). RT Redirect (rt-redirect) sub-type 0x08

The redirect extended community allows the traffic to be redirected to a VRF routing instance that lists the specified route-target in its import policy. If several local instances match this criteria, the choice between them is a local matter (for example, the instance with the lowest Route Distinguisher value can be elected).

This Extended Community allows 3 different encodings formats for the route-target (type 0x80, 0x81, 0x82). It uses the same encoding as the Route Target Extended Community in Sections [3.1](#) (type 0x80: 2-octet AS, 4-octet value), [3.2](#) (type 0x81: 4-octet IPv4 address, 2-octet value) and 4 of [RFC4360](#) and [Section 2](#) (type 0x82: 4-octet



AS, 2-octet value) of [[RFC5668](#)] with the high-order octet of the Type field 0x80, 0x81, 0x82 respectively and the low-order of the Type field (Sub-Type) always 0x08.

Interferes with: No other BGP Flow Specification Traffic Filtering Action in this document.

### **7.5. Traffic Marking (traffic-marking) sub-type 0x09**

The traffic marking extended community instructs a system to modify the DSCP bits in the IP header ([\[RFC2474\] Section 3](#)) of a transiting IP packet to the corresponding value encoded in the 6 least significant bits of the extended community value as shown in Figure 6.

The extended is encoded as follows:

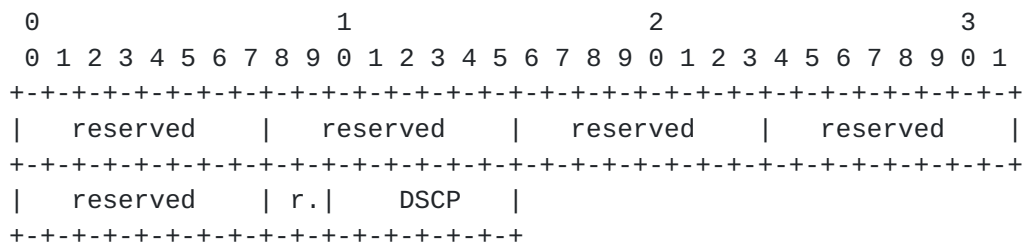


Figure 6: Traffic Marking Extended Community Encoding

- o DSCP: new DSCP value for the transiting IP packet.
- o reserved, r.: MUST be set to 0 on encoding, and MUST be ignored during decoding.

Interferes with: No other BGP Flow Specification Traffic Filtering Action in this document.

### **7.6. Interaction with other Filtering Mechanisms in Routers**

Implementations should provide mechanisms that map an arbitrary BGP community value (normal or extended) to Traffic Filtering Actions that require different mappings on different systems in the network. For instance, providing packets with a worse-than-best-effort per-hop behavior is a functionality that is likely to be implemented differently in different systems and for which no standard behavior is currently known. Rather than attempting to define it here, this can be accomplished by mapping a user-defined community value to platform-/network-specific behavior via user configuration.



### **7.7. Considerations on Traffic Filtering Action Interference**

Since Traffic Filtering Actions are represented as BGP extended community values, Traffic Filtering Actions may interfere with each other (e.g. there may be more than one conflicting traffic-rate-bytes Traffic Filtering Action associated with a single Flow Specification). Traffic Filtering Action interference has no impact on BGP propagation of Flow Specifications (all communities are propagated according to policies).

If a Flow Specification associated with interfering Traffic Filtering Actions is selected for packet forwarding, it is an implementation decision which of the interfering Traffic Filtering Actions are selected. Implementors of this specification **SHOULD** document the behaviour of their implementation in such cases.

Operators are encouraged to make use of the BGP policy framework supported by their implementation in order to achieve a predictable behaviour. See also [Section 12](#).

## **8. Dissemination of Traffic Filtering in BGP/MPLS VPN Networks**

Provider-based Layer 3 VPN networks, such as the ones using a BGP/MPLS IP VPN [[RFC4364](#)] control plane, may have different traffic filtering requirements than Internet service providers. But also Internet service providers may use those VPNs for scenarios like having the Internet routing table in a VRF, resulting in the same traffic filtering requirements as defined for the global routing table environment within this document. This document defines an additional BGP NLRI type (AFI=1, SAFI=134) value, which can be used to propagate Flow Specification in a BGP/MPLS VPN environment.

The NLRI format for this address family consists of a fixed-length Route Distinguisher field (8 octets) followed by the Flow Specification NLRI value ([Section 4.2](#)). The NLRI length field shall include both the 8 octets of the Route Distinguisher as well as the subsequent Flow Specification NLRI value. The resulting encoding is shown in Figure 7.

```

+-----+
| length (0xnn or 0xfn nn)      |
+-----+
| Route Distinguisher (8 octets) |
+-----+
|   NLRI value  (variable)      |
+-----+

```

Figure 7: Flow Specification NLRI for MPLS



Propagation of this NLRI is controlled by matching Route Target extended communities associated with the BGP path advertisement with the VRF import policy, using the same mechanism as described in BGP/MPLS IP VPNs [[RFC4364](#)].

Flow Specifications received via this NLRI apply only to traffic that belongs to the VRF(s) in which it is imported. By default, traffic received from a remote PE is switched via an MPLS forwarding decision and is not subject to filtering.

Contrary to the behavior specified for the non-VPN NLRI, Flow Specifications are accepted by default, when received from remote PE routers.

The validation procedure ([Section 6](#)) and Traffic Filtering Actions ([Section 7](#)) are the same as for IPv4.

## **[9.](#) Traffic Monitoring**

Traffic filtering applications require monitoring and traffic statistics facilities. While this is an implementation specific choice, implementations SHOULD provide:

- o A mechanism to log the packet header of filtered traffic.
- o A mechanism to count the number of matches for a given Flow Specification.

## **[10.](#) Error Handling**

Error handling according to [[RFC7606](#)] and [[RFC4760](#)] applies to this specification.

This document introduces Traffic Filtering Action Extended Communities. Malformed Traffic Filtering Action Extended Communities in the sense of [[RFC7606](#)] [Section 7.14](#). are Extended Community values that cannot be decoded according to [Section 7](#) of this document.

## **[11.](#) IANA Considerations**

This section complies with [[RFC7153](#)].

### **[11.1.](#) AFI/SAFI Definitions**

IANA maintains a registry entitled "SAFI Values". For the purpose of this work, IANA is requested to update the following SAFIs to read according to the table below (Note: This document obsoletes both





[RFC7674](#) and [RFC5575](#) and all references to those documents should be deleted from the registry below):

Value	Name	Reference
133	Dissemination of Flow Specification rules	[this document]
134	L3VPN Dissemination of Flow Specification rules	[this document]

Table 3: Registry: SAFI Values

The above textual changes generalise the definition of the SAFIs rather than change its underlying meaning. Therefore, based on "The YANG 1.1 Data Modeling Language" [[RFC7950](#)], the above text implies that the following YANG enums from "Common YANG Data Types for the Routing Area" [[RFC8294](#)] need to have their names and descriptions at <https://www.iana.org/assignments/iana-routing-types> [2] changed to:

```
<CODE BEGINS>
  enum flow-spec-safi {
    value 133;
    description
      "Dissemination of Flow Specification rules SAFI.";
  }
  enum l3vpn-flow-spec-safi {
    value 134;
    description
      "L3VPN Dissemination of Flow Specification rules SAFI.";
  }
<CODE ENDS>
```

A new revision statement should be added to the module as follows:

```
<CODE BEGINS>
  revision [revision date] {
    description "Non-backwards-compatible change of SAFI names
      (SAFI values 133, 134).";
    reference
      "[this document]: Dissemination of Flow Specification Rules.";
  }
<CODE ENDS>
```



## 11.2. Flow Component Definitions

A Flow Specification consists of a sequence of flow components, which are identified by an 8-bit component type. IANA has created and maintains a registry entitled "Flow Spec Component Types". IANA is requested to update the reference for this registry to [this document]. Furthermore the references to the values should be updated according to the table below (Note: This document obsoletes both [RFC7674](#) and [RFC5575](#) and all references to those documents should be deleted from the registry below).

Value	Name	Reference
1	Destination Prefix	[this document]
2	Source Prefix	[this document]
3	IP Protocol	[this document]
4	Port	[this document]
5	Destination port	[this document]
6	Source port	[this document]
7	ICMP type	[this document]
8	ICMP code	[this document]
9	TCP flags	[this document]
10	Packet length	[this document]
11	DSCP	[this document]
12	Fragment	[this document]

Table 4: Registry: Flow Spec Component Types

In order to manage the limited number space and accommodate several usages, the following policies defined by [RFC8126](#) are used:

Type Values	Policy
0	Reserved
[1 .. 127]	Specification Required
[128 .. 254]	Expert Review
255	Reserved

Table 5: Flow Spec Component Types Policies

### Guidance for Experts:

128-254 requires Expert Review as the registration policy. The Experts are expected to check the clarity of purpose and use of the requested code points. The Experts must also verify that



any specification produced in the IETF that requests one of these code points has been made available for review by the IDR working group and that any specification produced outside the IETF does not conflict with work that is active or already published within the IETF. It must be pointed out that introducing new component types may break interoperability with existing implementations of this protocol.

### **11.3. Extended Community Flow Specification Actions**

The Extended Community Flow Specification Action types defined in this document consist of two parts:

Type (BGP Transitive Extended Community Type)

Sub-Type

For the type-part, IANA maintains a registry entitled "BGP Transitive Extended Community Types". For the purpose of this work ([Section 7](#)), IANA is requested to update the references to the following entries according to the table below (Note: This document obsoletes both [RFC7674](#) and [RFC5575](#) and all references to those documents should be deleted in the registry below):

Type	Name	Reference
Value		
0x81	Generic Transitive Experimental Use Extended Community Part 2 (Sub-Types are defined in the "Generic Transitive Experimental Use Extended Community Part 2 Sub-Types" Registry)	[this document]
0x82	Generic Transitive Experimental Use Extended Community Part 3 (Sub-Types are defined in the "Generic Transitive Experimental Use Extended Community Part 3 Sub-Types" Registry)	[this document]

Table 6: Registry: BGP Transitive Extended Community Types

For the sub-type part of the extended community Traffic Filtering Actions IANA maintains the following registries. IANA is requested to update all names and references according to the tables below and assign a new value for the "Flow spec traffic-rate-packets" Sub-Type (Note: This document obsoletes both [RFC7674](#) and [RFC5575](#) and all



references to those documents should be deleted from the registries below).

Sub-Type Value	Name	Reference
0x06	Flow spec traffic-rate-bytes	[this document]
TBD	Flow spec traffic-rate-packets	[this document]
0x07	Flow spec traffic-action (Use of the "Value" field is defined in the "Traffic Action Fields" registry)	[this document]
0x08	Flow spec rt-redirect AS-2octet format	[this document]
0x09	Flow spec traffic-remarking	[this document]

Table 7: Registry: Generic Transitive Experimental Use Extended Community Sub-Types

Sub-Type Value	Name	Reference
0x08	Flow spec rt-redirect IPv4 format	[this document]

Table 8: Registry: Generic Transitive Experimental Use Extended Community Part 2 Sub-Types

Sub-Type Value	Name	Reference
0x08	Flow spec rt-redirect AS-4octet format	[this document]

Table 9: Registry: Generic Transitive Experimental Use Extended Community Part 3 Sub-Types

Furthermore IANA is requested to update the reference for the registries "Generic Transitive Experimental Use Extended Community





Part 2 Sub-Types" and "Generic Transitive Experimental Use Extended Community Part 3 Sub-Types" to [this document].

The "traffic-action" extended community ([Section 7.3](#)) defined in this document has 46 unused bits, which can be used to convey additional meaning. IANA created and maintains a registry entitled: "Traffic Action Fields". IANA is requested to update the reference for this registry to [this document]. Furthermore IANA is requested to update the references according to the table below. These values should be assigned via IETF Review rules only (Note: This document obsoletes both [RFC7674](#) and [RFC5575](#) and all references to those documents should be deleted from the registry below).

Bit	Name	Reference
47	Terminal Action	[this document]
46	Sample	[this document]

Table 10: Registry: Traffic Action Fields

## 12. Security Considerations

As long as Flow Specifications are restricted to match the corresponding unicast routing paths for the relevant prefixes ([Section 6](#)), the security characteristics of this proposal are equivalent to the existing security properties of BGP unicast routing. Any relaxation of the validation procedure described in [Section 6](#) may allow unwanted Flow Specifications to be propagated and thus unwanted Traffic Filtering Actions may be applied to flows.

Where the above mechanisms are not in place, this could open the door to further denial-of-service attacks such as unwanted traffic filtering, remarking or redirection.

Deployment of specific relaxations of the validation within an administrative boundary of a network are useful in some networks for quickly distributing filters to prevent denial-of-service attacks. For a network to utilize this relaxation, the BGP policies must support additional filtering since the origin AS field is empty. Specifications relaxing the validation restrictions MUST contain security considerations that provide details on the required additional filtering. For example, the use of Origin validation can provide enhanced filtering within an AS confederation.

Inter-provider routing is based on a web of trust. Neighboring autonomous systems are trusted to advertise valid reachability



information. If this trust model is violated, a neighboring autonomous system may cause a denial-of-service attack by advertising reachability information for a given prefix for which it does not provide service (unfiltered address space hijack). Since validation of the Flow Specification is tied to the announcement of the best unicast route, the failure in the validation of best path route may prevent the Flow Specification from being used by a local router. Possible mitigations are [[RFC6811](#)] and [[RFC8205](#)].

On IXPs routes are often exchanged via route servers which do not extend the AS\_PATH. In such cases it is not possible to enforce the left-most AS in the AS\_PATH to be the neighbor AS (the AS of the route server). Since the validation of Flow Specification ([Section 6](#)) depends on this, additional care must be taken. It is advised to use a strict inbound route policy in such scenarios.

Enabling firewall-like capabilities in routers without centralized management could make certain failures harder to diagnose. For example, it is possible to allow TCP packets to pass between a pair of addresses but not ICMP packets. It is also possible to permit packets smaller than 900 or greater than 1000 octets to pass between a pair of addresses, but not packets whose length is in the range 900- 1000. Such behavior may be confusing and these capabilities should be used with care whether manually configured or coordinated through the protocol extensions described in this document.

Flow Specification BGP speakers (e.g. automated DDoS controllers) not properly programmed, algorithms that are not performing as expected, or simply rogue systems may announce unintended Flow Specifications, send updates at a high rate or generate a high number of Flow Specifications. This may stress the receiving systems, exceed their capacity, or lead to unwanted Traffic Filtering Actions being applied to flows.

While the general verification of the Flow Specification NLRI is specified in this document ([Section 6](#)) the Traffic Filtering Actions received by a third party may need custom verification or filtering. In particular all non traffic-rate actions may allow a third party to modify packet forwarding properties and potentially gain access to other routing-tables/VPNs or undesired queues. This can be avoided by proper filtering/screening of the Traffic Filtering Action communities at network borders and only exposing a predefined subset of Traffic Filtering Actions (see [Section 7](#)) to third parties. One way to achieve this is by mapping user-defined communities, that can be set by the third party, to Traffic Filtering Actions and not accepting Traffic Filtering Action extended communities from third parties.



This extension adds additional information to Internet routers. These are limited in terms of the maximum number of data elements they can hold as well as the number of events they are able to process in a given unit of time. Service providers need to consider the maximum capacity of their devices and may need to limit the number of Flow Specifications accepted and processed.

### **13. Contributors**

Barry Greene, Pedro Marques, Jared Mauch and Nischal Sheth were authors on [[RFC5575](#)], and therefore are contributing authors on this document.

### **14. Acknowledgements**

The authors would like to thank Yakov Rekhter, Dennis Ferguson, Chris Morrow, Charlie Kaufman, and David Smith for their comments for the comments on the original [[RFC5575](#)]. Chaitanya Kodeboyina helped design the flow validation procedure; and Steven Lin and Jim Washburn ironed out all the details necessary to produce a working implementation in the original [[RFC5575](#)].

A packet rate Traffic Filtering Action was also described in a Flow Specification extension draft and the authors like to thank Wesley Eddy, Justin Dailey and Gilbert Clark for their work.

Additionally, the authors would like to thank Alexander Mayrhofer, Nicolas Fevrier, Job Snijders, Jeffrey Haas and Adam Chappell for their comments and review.

### **15. References**

#### **15.1. Normative References**

- [IEEE.754.1985]  
IEEE, "Standard for Binary Floating-Point Arithmetic",  
IEEE 754-1985, August 1985.
- [ISO\_IEC\_9899]  
ISO, "Information technology -- Programming languages --  
C", ISO/IEC 9899:2018, June 2018.
- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, [RFC 768](#),  
DOI 10.17487/RFC0768, August 1980,  
<<https://www.rfc-editor.org/info/rfc768>>.



- [RFC0791] Postel, J., "Internet Protocol", STD 5, [RFC 791](#), DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, [RFC 792](#), DOI 10.17487/RFC0792, September 1981, <<https://www.rfc-editor.org/info/rfc792>>.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", [RFC 2474](#), DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", [RFC 4271](#), DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", [RFC 4360](#), DOI 10.17487/RFC4360, February 2006, <<https://www.rfc-editor.org/info/rfc4360>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", [RFC 4364](#), DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", [RFC 4456](#), DOI 10.17487/RFC4456, April 2006, <<https://www.rfc-editor.org/info/rfc4456>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", [RFC 4760](#), DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.





- [RFC5668] Rekhter, Y., Sangli, S., and D. Tappan, "4-Octet AS Specific BGP Extended Community", [RFC 5668](#), DOI 10.17487/RFC5668, October 2009, <<https://www.rfc-editor.org/info/rfc5668>>.
- [RFC7153] Rosen, E. and Y. Rekhter, "IANA Registries for BGP Extended Communities", [RFC 7153](#), DOI 10.17487/RFC7153, March 2014, <<https://www.rfc-editor.org/info/rfc7153>>.
- [RFC7606] Chen, E., Ed., Scudder, J., Ed., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", [RFC 7606](#), DOI 10.17487/RFC7606, August 2015, <<https://www.rfc-editor.org/info/rfc7606>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 8126](#), DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## **15.2. Informative References**

- [I-D.ietf-idr-flow-spec-v6]  
Loibl, C., Raszuk, R., and S. Hares, "Dissemination of Flow Specification Rules for IPv6", [draft-ietf-idr-flow-spec-v6-15](#) (work in progress), September 2020.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", [RFC 4303](#), DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", [RFC 5575](#), DOI 10.17487/RFC5575, August 2009, <<https://www.rfc-editor.org/info/rfc5575>>.
- [RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", [RFC 6811](#), DOI 10.17487/RFC6811, January 2013, <<https://www.rfc-editor.org/info/rfc6811>>.
- [RFC7674] Haas, J., Ed., "Clarification of the Flowspec Redirect Extended Community", [RFC 7674](#), DOI 10.17487/RFC7674, October 2015, <<https://www.rfc-editor.org/info/rfc7674>>.



- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", [RFC 7950](#), DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8205] Lepinski, M., Ed. and K. Sriram, Ed., "BGPsec Protocol Specification", [RFC 8205](#), DOI 10.17487/RFC8205, September 2017, <<https://www.rfc-editor.org/info/rfc8205>>.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", [RFC 8294](#), DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.

### 15.3. URIs

- [1] <https://github.com/stoffi92/rfc5575bis/tree/master/flowspec-cmp>
- [2] <https://www.iana.org/assignments/iana-routing-types>

### Appendix A. Example Python code: flow\_rule\_cmp

<CODE BEGINS>

|| || ||

Copyright (c) 2020 IETF Trust and the persons identified as authors of [draft-ietf-idr-rfc5575bis](#). 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>).

|| || ||

```
import itertools
import collections
import ipaddress
```

EQUAL = 0

```
A_HAS_PRECEDENCE = 1
```

B HAS PRECEDENCE = 2

```
IP DESTINATION = 1
```

IP SOURCE = 2

[illegible]



```
class FS_nlri(object):
    """
    FS_nlri class implementation that allows sorting.

    By calling .sort() on a array of FS_nlri objects these will be
    sorted according to the flow_rule_cmp algorithm.

    Example:
    nlri = [ FS_nlri(components=[
        FS_component(component_type=IP_DESTINATION,
            op_value=ipaddress.ip_network('10.1.0.0/16') ),
        FS_component(component_type=4,
            op_value=bytearray([0,1,2,3,4,5,6])),
    ]),
        FS_nlri(components=[
            FS_component(component_type=5,
                op_value=bytearray([0,1,2,3,4,5,6])),
            FS_component(component_type=6,
                op_value=bytearray([0,1,2,3,4,5,6])),
        ]),
    ]
    nlri.sort() # sorts the array accorinding to the algorithm
    """
    def __init__(self, components = None):
        """
        components: list of type FS_component
        """
        self.components = components

    def __lt__(self, other):
        # use the below algorithm for sorting
        result = flow_rule_cmp(self, other)
        if result == B_HAS_PRECEDENCE:
            return True
        else:
            return False

def flow_rule_cmp(a, b):
    """
    Example of the flowspec comparison algorithm.
    """
    for comp_a, comp_b in itertools.zip_longest(a.components,
                                                b.components):
        # If a component type does not exist in one rule
        # this rule has lower precedence
        if not comp_a:
            return B_HAS_PRECEDENCE
```



```
    if not comp_b:
        return A_HAS_PRECEDENCE
    # Higher precedence for lower component type
    if comp_a.component_type < comp_b.component_type:
        return A_HAS_PRECEDENCE
    if comp_a.component_type > comp_b.component_type:
        return B_HAS_PRECEDENCE
    # component types are equal -> type specific comparison
    if comp_a.component_type in (IP_DESTINATION, IP_SOURCE):
        # assuming comp_a.op_value, comp_b.op_value of
        # type ipaddress.IPv4Network
        if comp_a.op_value.overlaps(comp_b.op_value):
            # longest prefixlen has precedence
            if comp_a.op_value.prefixlen > \
                comp_b.op_value.prefixlen:
                return A_HAS_PRECEDENCE
            if comp_a.op_value.prefixlen < \
                comp_b.op_value.prefixlen:
                return B_HAS_PRECEDENCE
            # components equal -> continue with next component
        elif comp_a.op_value > comp_b.op_value:
            return B_HAS_PRECEDENCE
        elif comp_a.op_value < comp_b.op_value:
            return A_HAS_PRECEDENCE
    else:
        # assuming comp_a.op_value, comp_b.op_value of type
        # bytearray
        if len(comp_a.op_value) == len(comp_b.op_value):
            if comp_a.op_value > comp_b.op_value:
                return B_HAS_PRECEDENCE
            if comp_a.op_value < comp_b.op_value:
                return A_HAS_PRECEDENCE
            # components equal -> continue with next component
        else:
            common = min(len(comp_a.op_value), len(comp_b.op_value))
            if comp_a.op_value[:common] > comp_b.op_value[:common]:
                return B_HAS_PRECEDENCE
            elif comp_a.op_value[:common] < \
                comp_b.op_value[:common]:
                return A_HAS_PRECEDENCE
            # the first common bytes match
            elif len(comp_a.op_value) > len(comp_b.op_value):
                return A_HAS_PRECEDENCE
            else:
                return B_HAS_PRECEDENCE

    return EQUAL
<CODE ENDS>
```





## **Appendix B. Comparison with [RFC 5575](#)**

This document includes numerous editorial changes to [[RFC5575](#)]. It also completely incorporates the redirect action clarification document [[RFC7674](#)]. It is recommended to read the entire document. The authors, however want to point out the following technical changes to [[RFC5575](#)]:

[Section 1](#) introduces the Flow Specification NLRI. In [[RFC5575](#)] this NLRI was defined as an opaque-key in BGPs database. This specification has removed all references to an opaque-key property. BGP implementations are able to understand the NLRI encoding.

[Section 4.2.1.1](#) defines a numeric operator and comparison bit combinations. In [[RFC5575](#)] the meaning of those bit combination was not explicitly defined and left open to the reader.

[Section 4.2.2.3](#) - [Section 4.2.2.8](#), [Section 4.2.2.10](#), [Section 4.2.2.11](#) make use of the above numeric operator. The allowed length of the comparison value was not consistently defined in [[RFC5575](#)].

[Section 7](#) defines all Traffic Filtering Action Extended communities as transitive extended communities. [[RFC5575](#)] defined the traffic-rate action to be non-transitive and did not define the transitivity of the other Traffic Filtering Action communities at all.

[Section 7.2](#) introduces a new Traffic Filtering Action (traffic-rate-packets). This action did not exist in [[RFC5575](#)].

[Section 7.4](#) contains the same redirect actions already defined in [[RFC5575](#)] however, these actions have been renamed to "rt-redirect" to make it clearer that the redirection is based on route-target. This section also completely incorporates the [[RFC7674](#)] clarifications of the Flowspec Redirect Extended Community.

[Section 7.7](#) contains general considerations on interfering traffic actions. [Section 7.3](#) also cross-references [Section 7.7](#). [[RFC5575](#)] did not mention this.

[Section 10](#) contains new error handling.



## Authors' Addresses

Christoph Loibl  
next layer Telekom GmbH  
Mariahilfer Guertel 37/7  
Vienna 1150  
AT

Phone: +43 664 1176414  
Email: cl@tix.at

Susan Hares  
Huawei  
7453 Hickory Hill  
Saline, MI 48176  
USA

Email: shares@ndzh.com

Robert Raszuk  
Bloomberg LP  
731 Lexington Ave  
New York City, NY 10022  
USA

Email: robert@raszuk.net

Danny McPherson  
Verisign  
USA

Email: dmcpherson@verisign.com

Martin Bacher  
T-Mobile Austria  
Rennweg 97-99  
Vienna 1030  
AT

Email: mb.ietf@gmail.com

