

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 10, 2020

P. Pfister  
E. Vyncke  
Cisco  
T. Pauly  
Apple Inc.  
D. Schinazi  
Google LLC  
W. Shao  
Cisco  
October 08, 2019

Discovering Provisioning Domain Names and Data  
draft-ietf-intarea-provisioning-domains-08

Abstract

Provisioning Domains (PvDs) are defined as consistent sets of network configuration information. This allows hosts to manage connections to multiple networks and interfaces simultaneously, such as when a home router provides connectivity through both a broadband and cellular network provider.

This document defines a mechanism for explicitly identifying PvDs through a Router Advertisement (RA) option. This RA option announces a PvD identifier, which hosts can compare to differentiate between PvDs. The option can directly carry some information about a PvD and can optionally point to additional PvD information that can be retrieved using HTTP over TLS.

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 10, 2020.

## Copyright Notice

Copyright (c) 2019 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

1. Introduction . . . . .	3
1.1. Specification of Requirements . . . . .	4
2. Terminology . . . . .	4
3. Provisioning Domain Identification using Router Advertisements . . . . .	5
3.1. PvD ID Option for Router Advertisements . . . . .	5
3.2. Router Behavior . . . . .	8
3.3. Non-PvD-aware Host Behavior . . . . .	9
3.4. PvD-aware Host Behavior . . . . .	9
3.4.1. DHCPv6 configuration association . . . . .	10
3.4.2. DHCPv4 configuration association . . . . .	10
3.4.3. Connection Sharing by the Host . . . . .	11
3.4.4. Usage of DNS Servers . . . . .	12
4. Provisioning Domain Additional Information . . . . .	12
4.1. Retrieving the PvD Additional Information . . . . .	13
4.2. Operational Consideration to Providing the PvD Additional Information . . . . .	15
4.3. PvD Additional Information Format . . . . .	15
4.3.1. Example . . . . .	17
4.4. Detecting misconfiguration and misuse . . . . .	17
5. Operational Considerations . . . . .	18
5.1. Exposing Extra RA Options to PvD-Aware Hosts . . . . .	18
5.2. Different RAs for PvD-Aware and Non-PvD-Aware Hosts . . . . .	18
5.3. Enabling Multi-homing for PvD-Aware Hosts . . . . .	20
6. Security Considerations . . . . .	21
7. Privacy Considerations . . . . .	21
8. IANA Considerations . . . . .	22
8.1. New entry in the Well-Known URIs Registry . . . . .	22
8.2. Additional Information PvD Keys Registry . . . . .	22
8.3. PvD Option Flags Registry . . . . .	22
8.4. PvD JSON Media Type Registration . . . . .	23

9. Acknowledgments . . . . .	23
10. References . . . . .	24
10.1. Normative References . . . . .	24
10.2. Informative References . . . . .	25
Authors' Addresses . . . . .	27

## 1. Introduction

Provisioning Domains (PvDs) are defined in [RFC7556] as consistent sets of network configuration information. This information includes properties that are traditionally associated with a single networking interface, such as source addresses, DNS configuration, proxy configuration, and gateway addresses.

Clients that are aware of PvDs can take advantage of multiple network interfaces simultaneously. This enables using two PvDs in parallel for separate connections or for multi-path transports.

While most PvDs today are discovered implicitly (such as by receiving information via Router Advertisements from a router on a network that a client host directly connects to), [RFC7556] also defines the notion of Explicit PvDs. IPsec Virtual Private Networks are considered Explicit PvDs, but Explicit PvDs can also be discovered via the local network router. Discovering Explicit PvDs allows two key advancements in managing multiple PvDs:

1. The ability to discover and use multiple PvDs on a single interface, such as when a local router can provide connectivity to two different Internet Service Providers.
2. The ability to associate additional informations about PvDs to describe the properties of the network.

While [RFC7556] defines the concept of Explicit PvDs, it does not define the mechanism for discovering multiple Explicit PvDs on a single network and their additional information.

This document specifies a way to identify PvDs with Fully Qualified Domain Names (FQDN), called PvD IDs. Those identifiers are advertised in a new Router Advertisement (RA) [RFC4861] option called the PvD ID Router Advertisement option which, when present, associates the PvD ID with all the information present in the Router Advertisement as well as any configuration object, such as addresses, deriving from it. The PVD ID Router Advertisement option may also contain a set of other RA options. Since such options are only considered by hosts implementing this specification, network operators may configure hosts that are 'PvD-aware' with PvDs that are ignored by other hosts.

Since PvD IDs are used to identify different ways to access the internet, multiple PvDs (with different PvD IDs) can be provisioned on a single host interface. Similarly, the same PvD ID could be used on different interfaces of a host in order to inform that those PvDs ultimately provide equivalent services.

This document also introduces a mechanism for hosts to retrieve optional additional information related to a specific PvD by means of an HTTP over TLS query using an URI derived from the PvD ID. The retrieved JSON object contains additional information that would typically be considered too large to be directly included in the Router Advertisement, but might be considered useful to the applications, or even sometimes users, when choosing which PvD should be used.

For example, if Alice has both a cellular network provider and a broadband provider in her home, her PvD-aware devices and applications would be aware of both available uplinks. These applications could fail-over between these networks, or run connections over both (potentially using multi-path transports). Applications could also select specific uplinks based on the properties of the network; for example, if the cellular network provides free high-quality video streaming, a video-streaming application could select that network while most of the other traffic on Alice's device uses the broadband provider.

### 1.1. Specification of Requirements

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.

## 2. Terminology

This document uses the following terminology:

**Provisioning Domain (PvD):** A set of network configuration information; for more information, see [RFC7556].

**PvD ID:** A Fully Qualified Domain Name (FQDN) used to identify a PvD.

**Explicit PvD:** A PvD uniquely identified with a PvD ID. For more information, see [RFC7556].

Implicit PvD: A PvD that, in the absence of a PvD ID, is identified by the host interface to which it is attached and the address of the advertising router. See also [RFC7556].

PvD-aware host: A host that supports the association of network configuration information into PvDs and the use of these PvDs as described in this document. Also named PvD-aware node in [RFC7556].

### 3. Provisioning Domain Identification using Router Advertisements

Explicit PvDs are identified by a PvD ID. The PvD ID is a Fully Qualified Domain Name (FQDN) which MUST belong to the network operator in order to avoid naming collisions. The same PvD ID MAY be used in several access networks when they ultimately provide identical services (e.g., in all home networks subscribed to the same service); else, the PvD ID MUST be different to follow Section 2.4 of [RFC7556].

#### 3.1. PvD ID Option for Router Advertisements

This document introduces a Router Advertisement (RA) option called PvD Option. It is used to convey the FQDN identifying a given PvD (see Figure 1), bind the PvD ID with configuration information received over DHCPv4 (see Section 3.4.2), enable the use of HTTP over TLS to retrieve the PvD Additional Information JSON object (see Section 4), as well as contain any other RA options which would otherwise be valid in the RA.

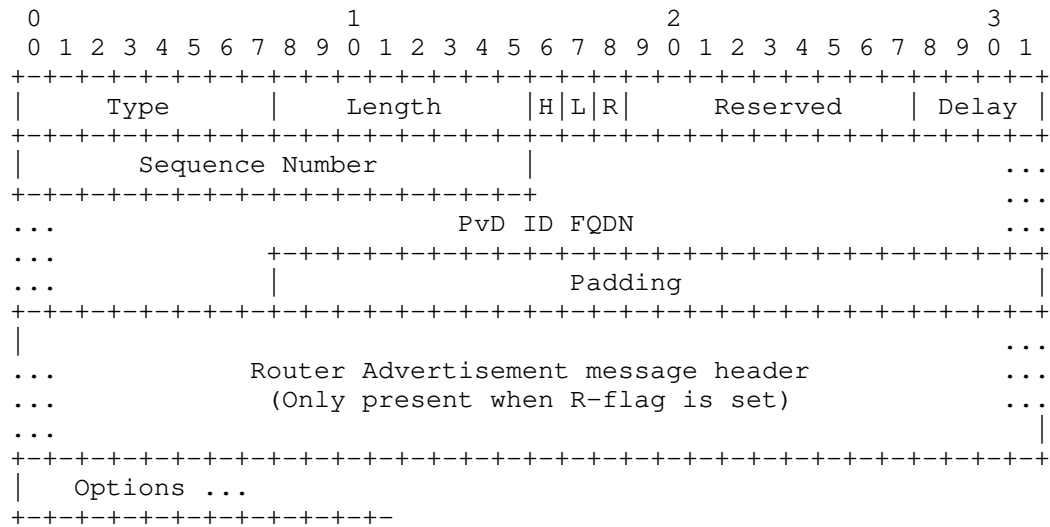


Figure 1: PvD ID Router Advertisements Option Format

Type: (8 bits) Set to 21.

Length: (8 bits) The length of the option in units of 8 octets, including the Type and Length fields, the Router Advertisement message header, if any, as well as the RA options that are included within the PvD Option.

H-flag: (1 bit) 'HTTP' flag stating whether some PvD Additional Information is made available through HTTP over TLS, as described in Section 4.

L-flag: (1 bit) 'Legacy' flag stating whether the router is also providing IPv4 information using DHCPv4 (see Section 3.4.2).

R-flag: (1 bit) 'Router Advertisement' flag stating whether the PvD Option is followed (right after padding to the next 64 bits boundary) by a Router Advertisement message header (See section 4.2 of [RFC4861]).

Reserved: (13 bits) Reserved for later use. It MUST be set to zero by the sender and ignored by the receiver.

Delay: (4 bits) Unsigned integer used to delay HTTP GET queries from hosts by a randomized backoff (see Section 4.1).

Sequence Number: (16 bits) Sequence number for the PvD Additional Information, as described in Section 4.

PvD ID FQDN: The FQDN used as PvD ID encoded in DNS format, as described in Section 3.1 of [RFC1035]. Domain names compression described in Section 4.1.4 of [RFC1035] MUST NOT be used.

Padding: Zero or more padding octets to the next 8 octet boundary (see Section 4.6 of [RFC4861]). It MUST be set to zero by the sender, and ignored by the receiver.

RA message header: (16 octets) When the R-flag is set, a full Router Advertisement message header as specified in [RFC4861]. The sender MUST set the 'Type' to 134, the value for "Router Advertisement", and set the 'Code' to 0. Receivers MUST ignore both of these fields. The 'Checksum' MUST be set to 0 by the sender; non-zero checksums MUST be ignored by the receiver. All other fields are to be set and parsed as specified in [RFC4861] or any updating documents.

Options: Zero or more RA options that would otherwise be valid as part of the Router Advertisement main body, but are instead included in the PvD Option so as to be ignored by hosts that are not PvD-aware.

Here is an example of a PvD Option with "example.org" as the PvD ID FQDN and including both an RDNSS option and a prefix information option. It has a Sequence Number of 123, and indicates the presence of additional information that is expected to be fetched with a delay factor of 5.

0										1										2										3																			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																		
Type: 21										Length: 12										1 0 0										Reserved										Delay:5									
Seq number: 123										7										e																													
x										a										m										p																			
l										e										3										o																			
r										g										0										0 (padding)																			
0 (padding)										0 (padding)										0 (padding)										0 (padding)																			
RDNSS option (RFC 6106) length: 5																														...																			
...																														...																			
...																																																	
Prefix Information Option (RFC 4861) length: 4																														...																			
...																																																	
...																																																	

Figure 2

### 3.2. Router Behavior

A router MAY send RAs containing one PvD Option, but MUST NOT include more than one PvD Option in each RA. The PvD Option MUST NOT contain further PvD Options.

The PvD Option MAY contain zero, one, or more RA options which would otherwise be valid as part of the same RA. Such options are processed by PvD-aware hosts, while ignored by other hosts per section 4.2 of [RFC4861].

In order to provide multiple different PvDs, a router MUST send multiple RAs. If more than one different Implicit PvDs are advertised, the RAs MUST be sent from different link-local source addresses. Explicit PvDs MAY share link-local source addresses with an Implicit PvD and any number of other Explicit PvDs.

In other words, different Explicit PvDs MAY be advertised with RAs using the same link-local source address; but different Implicit PvDs, advertised by different RAs, MUST use different link-local addresses because these Implicit PvDs are identified by the source addresses of the RAs.



As specified in [RFC4861], when the set of options causes the size of an advertisement to exceed the link MTU, multiple router advertisements can be sent, each containing a subset of the options. In such cases, the PvD Option header (i.e., all fields except the 'Options' field) MUST be repeated in all the transmitted RAs. The options within the 'Options' field, MAY be transmitted only once, included in one of the transmitted PvD Options.

### 3.3. Non-PvD-aware Host Behavior

As the PvD Option has a new option code, non-PvD-aware hosts will simply ignore the PvD Option and all the options it contains (see section 4.2 of [RFC4861]). This ensure the backward compatibility required in Section 3.3 of [RFC7556]. This behavior allows for a mixed-mode network with a mix of PvD-aware and non-PvD-aware hosts coexist.

### 3.4. PvD-aware Host Behavior

Hosts MUST associate received RAs and included configuration information (e.g., Router Valid Lifetime, Prefix Information [RFC4861], Recursive DNS Server [RFC8106], Routing Information [RFC4191] options) with the Explicit PvD identified by the first PvD Option present in the received RA, if any, or with the Implicit PvD identified by the host interface and the source address of the received RA otherwise.

In case multiple PvD Options are found in a given RA, hosts MUST ignore all but the first PvD Option.

If a host receives PvD Options flags that it does not recognize (currently in the Reserved field), it MUST ignore these flags.

Similarly, hosts MUST associate all network configuration objects (e.g., default routers, addresses, more specific routes, DNS Recursive Resolvers) with the PvD associated with the RA which last updated the object. For example, addresses that are generated using a received Prefix Information option (PIO) are associated with the PvD of the last received RA which included the given PIO.

PvD IDs MUST be compared in a case-insensitive manner as defined by [RFC4343]. For example, "pvd.example.com." or "PvD.Example.coM." would refer to the same PvD.

While resolving names, executing the default address selection algorithm [RFC6724] or executing the default router selection algorithm when forwarding packets ([RFC4861], [RFC4191] and

[RFC8028]), hosts and applications MAY consider only the configuration associated with any non-empty subset of PvDs.

For example, a host MAY associate a given process with a specific PvD, or a specific set of PvDs, while associating another process with another PvD. A PvD-aware application might also be able to select, on a per-connection basis, which PvDs should be used. In particular, constrained devices such as small battery operated devices (e.g. IoT), or devices with limited CPU or memory resources may purposefully use a single PvD while ignoring some received RAs containing different PvD IDs.

The way an application expresses its desire to use a given PvD, or a set of PvDs, or the way this selection is enforced, is out of the scope of this document. Useful insights about these considerations can be found in [I-D.kline-mif-mpvd-api-reqs].

#### 3.4.1. DHCPv6 configuration association

When a host retrieves stateless configuration elements using DHCPv6 (e.g., DNS recursive resolvers or DNS domain search lists [RFC3646]), they MUST be associated with all the explicit and implicit PvDs received on the same interface and contained in a RA with the O-flag set [RFC4861].

When a host retrieves stateful assignments using DHCPv6, such assignments MUST be associated with the received PvD which was received with RAs with the M-flag set and including a matching PIO. A PIO is considered to match a DHCPv6 assignment when the IPv6 prefix from the PIO includes the assignment from DHCPv6. For example, if a PvD's associated PIO defines the prefix 2001:db8:cafe::/64, a DHCPv6 IA\_NA message that assigns the address 2001:db8:cafe::1234:4567 would be considered to match.

In cases where an address would be assigned by DHCPv6 and no matching PvD could be found, hosts MAY associate the assigned address with any implicit PvD received on the same interface or to multiple of implicit PvD received on the same interface. This is intended to resolve backward compatibility issues with rare deployments choosing to assign addresses with DHCPv6 while not sending any matching PIO.

#### 3.4.2. DHCPv4 configuration association

Associating DHCPv4 [RFC2131] configuration elements with Explicit PvDs allows hosts to treat a set of IPv4 and IPv6 configurations as a single PvD with shared properties. For example, consider a router that provides two different uplinks. One could be a broadband network that has data rate and streaming properties described in PvD

additional information and that provides both IPv4 and IPv6 network access. The other could be a cellular network that provides only IPv6 network access, and uses NAT64 [RFC6146]. The broadband network can be represented by an Explicit PvD that points to the additional information, and also marks association with DHCPv4 information. The cellular network can be represented by a different Explicit PvD that is not associated with DHCPv4.

When a PvD-aware host retrieves configuration elements from DHCPv4, the information is associated either with a single Explicit PvD on that interface, or else with all Implicit PvDs on the same interface.

An Explicit PvD indicates its association with DHCPv4 information by setting the L-flag in the PvD RA Option. If there is exactly one Explicit PvD that sets this flag, hosts MUST associate the DHCPv4 information with that PvD. Multiple Explicit PvDs on the same interface marking this flag is a misconfiguration, and hosts SHOULD NOT associate the DHCPv4 information with any Explicit PvD in this case.

If no single Explicit PvD claims association with DHCPv4, the configuration elements coming from DHCPv4 MUST be associated with the all Implicit PvDs identified by the interface on which the DHCPv4 transaction happened. This maintains existing host behavior.

#### 3.4.3. Connection Sharing by the Host

The situation when a host shares connectivity from an upstream interface (e.g. cellular) to a downstream interface (e.g. Wi-Fi) is known as 'tethering'. Techniques such as ND-proxy [RFC4389], 64share [RFC7278] or prefix delegation (e.g. using DHCPv6-PD [RFC8415]) may be used for that purpose.

Whenever the RAs received from the upstream interface contain a PVD RA option, hosts that are sharing connectivity SHOULD include a PVD option within the RAs sent downstream with:

- o The same PVD-ID FQDN
- o The same H-bit, Delay and Sequence Number values
- o The L bit set whenever the host is sharing IPv4 connectivity received from the same upstream interface
- o The bits from the Reserved field set to 0

The values of the R-bit, Router Advertisement message header and Options field depend on whether the connectivity should be shared

only with PvD-aware hosts or not (see Section 3.2). In particular, all options received within the upstream PvD Option and included in the downstream RA SHOULD be included in the downstream PvD Option.

#### 3.4.4. Usage of DNS Servers

PvD-aware hosts can be provisioned with recursive DNS servers via RA options passed within an Explicit PvD, via RA options associated with an Implicit PvD, via DHCPv6 or DHCPv4, or from some other provisioning mechanism that creates an Implicit PvD (such as a VPN). In all of these cases, the DNS server addresses SHOULD be associated with the corresponding PvD. Specifically, queries sent to a configured recursive DNS server SHOULD be sent from a local IP address that was provisioned by the PvD via RA or DHCP. Answers received from the DNS server SHOULD only be used on the same PvD.

PvD-aware applications will be able to select which PvD(s) to use for DNS resolution and connections, which allows them to effectively use multiple Explicit PvDs. In order to support non-PvD-aware applications, however, PvD-aware hosts SHOULD ensure that non-PvD-aware name resolution APIs like "getaddrinfo" only use resolvers from a single PvD for each query. More discussion is provided in Section 5.2.1 of [RFC7556].

Maintaining the correct usage of DNS within PvDs avoids various practical errors, such as:

- o A PvD associated with a VPN or otherwise private network may provide DNS answers that contain addresses inaccessible over another PvD.
- o A PvD that uses a NAT64 [RFC6146] and DNS64 [RFC6147] will synthesize IPv6 addresses in DNS answers that are not globally routable, and would be invalid on other PvDs. Conversely, an IPv4 address resolved via DNS on another PvD cannot be directly used on a NAT64 network.

#### 4. Provisioning Domain Additional Information

Additional information about the network characteristics can be retrieved based on the PvD ID. This set of information is called PvD Additional Information, and is encoded as a JSON object [RFC8259]. This JSON object is restricted to the restricted profile of I-JSON, as defined in [RFC7493].

The purpose of this JSON object is to provide additional information to applications on a client host about the connectivity that is provided using a given interface and source address. It typically

includes data that would be considered too large, or not critical enough, to be provided within an RA option. The information contained in this object MAY be used by the operating system, network libraries, applications, or users, in order to decide which set of PvDs should be used for which connection, as described in Section 3.4.

The additional information related to a PvD is specifically intended to be optional, and is targeted at optimizing or informing the behavior of user-facing hosts. This information can be extended to provide hints for host system behavior (such as captive portal or walled-garden PvD detection) or application behavior (describing application-specific services offered on a given PvD). This content may not be appropriate for light-weight Internet of Things (IoT) devices. IoT devices might need only a subset of the information, and would in some cases prefer a smaller representation like CBOR ([RFC7049]). Delivering a reduced version of the PvD Additional Information designed for such devices is not defined in this document.

#### 4.1. Retrieving the PvD Additional Information

When the H-flag of the PvD Option is set, hosts MAY attempt to retrieve the PvD Additional Information associated with a given PvD by performing an HTTP over TLS [RFC2818] GET query to `https://<PvD-ID>/.well-known/pvd` [RFC8615]. Inversely, hosts MUST NOT do so whenever the H-flag is not set.

HTTP requests and responses for PvD additional information use the "application/pvd+json" media type (see Section 8). Clients SHOULD include this media type as an Accept header in their GET requests, and servers MUST mark this media type as their Content-Type header in responses.

Note that the DNS name resolution of the PvD ID, the PKI (Public Key Infrastructure) checks as well as the actual query MUST be performed using the considered PvD. In other words, the name resolution, PKI checks, source address selection, as well as the next-hop router selection MUST be performed while using exclusively the set of configuration information attached with the PvD, as defined in Section 3.4. In some cases, it may therefore be necessary to wait for an address to be available for use (e.g., once the Duplicate Address Detection or DHCPv6 processes are complete) before initiating the HTTP over TLS query. If the host has a temporary address per [RFC4941] in this PvD, then hosts SHOULD use a temporary address to fetch the PvD Additional Information and SHOULD deprecate the used temporary address and generate a new temporary address afterward.

If the HTTP status of the answer is greater than or equal to 400 the host MUST abandon and consider that there is no additional PvD information. If the HTTP status of the answer is between 300 and 399, inclusive, it MUST follow the redirection(s). If the HTTP status of the answer is between 200 and 299, inclusive, the host MAY get a file containing a single JSON object.

After retrieval of the PvD Additional Information, hosts MUST remember the last Sequence Number value received in the RA including the same PvD ID. Whenever a new RA for the same PvD is received with a different Sequence Number value, or whenever the expiry date for the additional information is reached, hosts MUST deprecate the additional information and stop using it until a new JSON object is retrieved.

Hosts retrieving a new PvD Additional Information object MUST check for the presence and validity of the mandatory fields specified in Section 4.3. A retrieved object including an expiration time that is already past or missing a mandatory element MUST be ignored.

In order to avoid synchronized queries toward the server hosting the PvD Additional Information when an object expires, object updates are delayed by a randomized backoff time.

- o When a host performs a JSON object update after it detected a change in the PvD Option Sequence Number, it MUST add a delay before sending the query. The target time for the delay is calculated as a random time between zero and  $2^{**}(\text{Delay} * 2)$  milliseconds, where 'Delay' corresponds to the 4-bit unsigned integer in the last received PvD Option.
- o When a host last retrieved a JSON object at time A that includes a expiry time B using the "expires" key, and the host is configured to keep the PvD information up to date, it MUST add some randomness into its calculation of the time to fetch the update. The target time for fetching the updated object is calculated as a uniformly random time in the interval  $[(B-A)/2, B]$ .

In the example Figure 2, the delay field value is 5, this means that host calculates its delay by choosing a random number between 0 and  $2^{**}(5 * 2)$  milliseconds, i.e., between 0 and 1024 milliseconds.

Since the 'Delay' value is directly within the PvD Option rather than the object itself, an operator may perform a push-based update by incrementing the Sequence value while changing the Delay value depending on the criticality of the update and its PvD Additional Information servers capacity.

The PvD Additional Information object includes a set of IPv6 prefixes (under the key "prefixes") which MUST be checked against all the Prefix Information Options advertised in the RA. If any of the prefixes included in any associated PIO is not covered by at least one of the listed prefixes, the associated PvD information MUST be considered to be a misconfiguration, and MUST NOT be used by the host. See Section 4.4 for more discussion on handling such misconfigurations.

#### 4.2. Operational Consideration to Providing the PvD Additional Information

Whenever the H-flag is set in the PvD Option, a valid PvD Additional Information object MUST be made available to all hosts receiving the RA by the network operator. In particular, when a captive portal is present, hosts MUST still be allowed to perform DNS, PKI and HTTP over TLS operations related to the retrieval of the object, even before logging into the captive portal.

Routers SHOULD increment the PVD Option Sequence Number by one whenever a new PvD Additional Information object is available and should be retrieved by hosts. If the value exceeds what can be stored in the Sequence Number field, it SHOULD wrap back to zero.

The server providing the JSON files SHOULD also check whether the client address is part of the prefixes listed into the additional information and SHOULD return a 403 response code if there is no match.

#### 4.3. PvD Additional Information Format

The PvD Additional Information is a JSON object.

The following table presents the mandatory keys which MUST be included in the object:

JSON key	Description	Type	Example
identifier	PvD ID FQDN	String	"pvd.example.com."
expires	Date after which this object is no longer valid	[RFC3339] Date	"2017-07-23T06:00:00Z"
prefixes	Array of IPv6 prefixes valid for this PvD	Array of strings	["2001:db8:1::/48", "2001:db8:4::/48"]

A retrieved object which does not include all three of these keys at the root of the JSON object MUST be ignored. All three keys need to be validated, otherwise the object MUST be ignored. The value stored for "identifier" MUST be matched against the PvD ID FQDN presented in the PvD RA option using the comparison mechanism described in Section 3.4. The value stored for "expires" MUST be a valid date in the future. If the PIO of the received RA is not covered by at least one of the "prefixes" key, the retrieved object SHOULD be ignored.

The following table presents some optional keys which MAY be included in the object.

JSON key	Description	Type	Example
dnsZones	DNS zones searchable and accessible	Array of strings	["example.com", "sub.example.com"]
noInternet	No Internet, set when the PvD is restricted.	Boolean	true

It is worth noting that the JSON format allows for extensions. Whenever an unknown key is encountered, it MUST be ignored along with its associated elements.

Private-use or experimental keys MAY be used in the JSON dictionary. In order to avoid such keys colliding with IANA registry keys, implementers or vendors defining private-use or experimental keys MUST create sub-dictionaries, where the sub-dictionary is added into



the top-level JSON dictionary with a key of the format "vendor-\*" where the "\*" is replaced by the implementer's or vendor's identifier. For example, keys specific to the FooBar organization could use "vendor-foobar". Upon receiving such a sub-dictionary, host MUST ignore this sub-dictionary if it is unknown. When the vendor or implementer is part of an IANA URN namespace [URN], the URN namespace SHOULD be used rather than the "vendor-\*" format.

#### 4.3.1. Example

The following two examples show how the JSON keys defined in this document can be used:

```
{
  "identifier": "cafe.example.com",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
}

{
  "identifier": "company.foo.example.com",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
  "vendor-foo": {
    "private-key": "private-value",
  },
}
```

#### 4.4. Detecting misconfiguration and misuse

When a host retrieves the PvD Additional Information, it MUST verify that the TLS server certificate is valid for the performed request (e.g., that the Subject Name is equal to the PvD ID expressed as an FQDN). This authentication creates a secure binding between the information provided by the trusted Router Advertisement, and the HTTPS server. However, this does not mean the Advertising Router and the PvD server belong to the same entity.

Hosts MUST verify that all prefixes in all the RA PIOs are covered by a prefix from the PvD Additional Information. An adversarial router attempting to spoof the definition of an Explicit PvD, without the ability to modify the PvD Additional Information, would need to perform NAT66 in order to circumvent this check. Thus, this check cannot prevent all spoofing, but it can detect misconfiguration or mismatched routers that are not adding a NAT.

If NAT66 is being added in order to spoof PvD ownership, the HTTPS server for additional information can detect this misconfiguration. The HTTPS server SHOULD validate the source addresses of incoming connections (see Section 4.1). This check gives reasonable assurance that neither NPTv6 [RFC6296] nor NAT66 were used and restricts the information to the valid network users. If the PvD does not provision IPv4 (it does not include the 'L' bit in the RA), the server cannot validate the source addresses of connections using IPv4. Thus, the PvD ID FQDN for such PvDs SHOULD NOT have a DNS A record.

## 5. Operational Considerations

This section describes some example use cases of PvD. For the sake of simplicity, the RA messages will not be described in the usual ASCII art but rather in an indented list.

### 5.1. Exposing Extra RA Options to PvD-Aware Hosts

In this example, there is one RA message sent by the router. This message contains some options applicable to all hosts on the network, and also a PvD Option that also contains other options only visible to PvD-aware hosts.

- o RA Header: router lifetime = 6000
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o PvD Option header: length = 3 + 5 + 4 , PvD ID FQDN = example.org., R-flag = 0 (actual length of the header with padding 24 bytes = 3 \* 8 bytes)
  - \* Recursive DNS Server: length = 5, addresses = [2001:db8:cafe::53, 2001:db8:f00d::53]
  - \* Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64

Note that a PvD-aware host will receive two different prefixes, 2001:db8:cafe::/64 and 2001:db8:f00d::/64, both associated with the same PvD (identified by "example.org."). A non-PvD-aware host will only receive one prefix, 2001:db8:cafe::/64.

### 5.2. Different RAs for PvD-Aware and Non-PvD-Aware Hosts

It is expected that for some years, networks will have a mixed environment of PvD-aware hosts and non-PvD-aware hosts. If there is a need to give specific information to PvD-aware hosts only, then it

is RECOMMENDED to send two RA messages, one for each class of hosts. If two RA messages are sent for this reason, they MUST be sent from two different link-local source addresses (Section 3.2). For example, here is the RA sent for non-PvD-aware hosts:

- o RA Header: router lifetime = 6000 (non-PvD-aware hosts will use this router as a default router)
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o Recursive DNS Server Option: length = 3, addresses=[2001:db8:cafe::53]
- o PvD Option header: length = 3 + 2, PvD ID FQDN = foo.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 \* 8 bytes)
  - \* RA Header: router lifetime = 0 (PvD-aware hosts will not use this router as a default router), implicit length = 2

And here is the RA sent for PvD-aware hosts:

- o RA Header: router lifetime = 0 (non-PvD-aware hosts will not use this router as a default router)
- o PvD Option header: length = 3 + 2 + 4 + 3, PvD ID FQDN = bar.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 \* 8 bytes)
  - \* RA Header: router lifetime = 1600 (PvD-aware hosts will use this router as a default router), implicit length = 2
  - \* Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64
  - \* Recursive DNS Server Option: length = 3, addresses = [2001:db8:f00d::53]

In the above example, non-PvD-aware hosts will only use the first RA sent from their default router and using the 2001:db8:cafe::/64 prefix. PvD-aware hosts will autonomously configure addresses from both PIOs, but will only use the source address in 2001:db8:f00d::/64 to communicate past the first hop router since only the router sending the second RA will be used as default router; similarly, they will use the DNS server 2001:db8:f00d::53 when communicating with this address.

### 5.3. Enabling Multi-homing for PvD-Aware Hosts

In this example, the goal is to have one prefix from one RA be usable by both non-PvD-aware and PvD-aware hosts; and to have another prefix usable only by PvD-aware hosts. This allows PvD-aware hosts to be able to effectively multi-home on the network.

The first RA is usable by all hosts. The only difference for PvD-aware hosts is that they can explicitly identify the PvD ID associated with the RA. PvD-aware hosts will also use this prefix to communicate with non-PvD-aware hosts on the same network.

- o RA Header: router lifetime = 6000 (non-PvD-aware hosts will use this router as a default router)
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o Recursive DNS Server Option: length = 3, addresses = [2001:db8:cafe::53]
- o PvD Option header: length = 3, PvD ID FQDN = foo.example.org., R-flag = 0 (actual length of the header 24 bytes = 3 \* 8 bytes)

The second RA contains a prefix usable only by PvD-aware hosts. Non-PvD-aware hosts will ignore this RA; hence, the only PvD-aware hosts will be multi-homed.

- o RA Header: router lifetime = 0 (non-PvD-aware hosts will not use this router as a default router)
- o PvD Option header: length = 3 + 2 + 4 + 3, PvD ID FQDN = bar.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 \* 8 bytes)
  - \* RA Header: router lifetime = 1600 (PvD-aware hosts will use this router as a default router), implicit length = 2
  - \* Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64
  - \* Recursive DNS Server Option: length = 3, addresses = [2001:db8:f00d::53]

Note: the above examples assume that the router has received its PvD IDs from upstream routers or via some other configuration mechanism. Another document could define ways for the router to generate its own PvD IDs to allow the above scenario in the absence of PvD ID provisioning.

## 6. Security Considerations

Although some solutions such as IPsec or SeND [RFC3971] can be used in order to secure the IPv6 Neighbor Discovery Protocol, in practice actual deployments largely rely on link layer or physical layer security mechanisms (e.g. 802.1x [IEEE8021X]) in conjunction with RA Guard [RFC6105].

This specification does not improve the Neighbor Discovery Protocol security model, but extends the purely link-local trust relationship between the host and the default routers with HTTP over TLS communications which servers are authenticated as rightful owners of the FQDN received within the trusted PvD ID RA option.

It must be noted that Section 4.4 of this document only provides reasonable assurance against misconfiguration but does not prevent an hostile network access provider to advertise wrong information that could lead applications or hosts to select a hostile PvD.

Users cannot be assumed to be able to meaningfully differentiate between "safe" and "unsafe" networks. This is a known attack surface that is present whether or not PvDs are in use, and hence cannot be addressed by this document. However, a host that correctly implements the multiple PvD architecture ([RFC7556]) using the mechanism described in this document will be less susceptible to such attacks than a host that does not by being able to check for the various misconfigurations described in this document.

## 7. Privacy Considerations

Retrieval of the PvD Additional Information over HTTPS requires early communications between the connecting host and a server which may be located further than the first hop router. Although this server is likely to be located within the same administrative domain as the default router, this property can't be ensured. Therefore, hosts willing to retrieve the PvD Additional Information before using it without leaking identity information, SHOULD make use of an IPv6 Privacy Address and SHOULD NOT include any privacy sensitive data, such as User Agent header or HTTP cookie, while performing the HTTP over TLS query.

From a privacy perspective, retrieving the PvD Additional Information is not different from establishing a first connection to a remote server, or even performing a single DNS lookup. For example, most operating systems already perform early queries to well known web sites, such as <http://captive.example.com/hotspot-detect.html>, in order to detect the presence of a captive portal.

There may be some cases where hosts, for privacy reasons, should refrain from accessing servers that are located outside a certain network boundary. In practice, this could be implemented as a whitelist of 'trusted' FQDNs and/or IP prefixes that the host is allowed to communicate with. In such scenarios, the host SHOULD check that the provided PvD ID, as well as the IP address that it resolves into, are part of the allowed whitelist.

## 8. IANA Considerations

Upon publication of this document, IANA is asked to remove the 'reclaimable' tag off the value 21 for the PvD Option (from the IPv6 Neighbor Discovery Option Formats registry).

### 8.1. New entry in the Well-Known URIs Registry

IANA is asked to add a new entry in the well-known-uris registry with the following information:

URI suffix: 'pvd'

Change controller: IETF

Specification document: this document

Status: permanent

Related information: N/A

### 8.2. Additional Information PvD Keys Registry

IANA is asked to create and maintain a new registry called "Additional Information PvD Keys", which will reserve JSON keys for use in PvD additional information. The initial contents of this registry are given in Section 4.3.

New assignments for Additional Information PvD Keys Registry will be administered by IANA through Expert Review [RFC8126].

### 8.3. PvD Option Flags Registry

IANA is also asked to create and maintain a new registry entitled "PvD Option Flags" reserving bit positions from 0 to 15 to be used in the PvD Option bitmask. Bit position 0, 1 and 2 are reserved by this document (as specified in Figure 1). Future assignments require Standards Action [RFC8126], via a Standards Track RFC document.

#### 8.4. PvD JSON Media Type Registration

This document registers the media type for PvD JSON text, "application/pvd+json".

Type Name: application

Subtype Name: pvd+json

Required parameters: None

Optional parameters: None

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type.

Security considerations: See Section 6.

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: This document

Applications that use this media type: This media type is intended to be used by network advertising additional Provisioning Domain information, and clients looking up such information.

Additional information: None

Person and email address to contact for further information: See Authors' Addresses section

Intended usage: COMMON

Restrictions on usage: None

Author: IETF

Change controller: IETF

#### 9. Acknowledgments

Many thanks to M. Stenberg and S. Barth for their earlier work: [I-D.stenberg-mif-mpvd-dns], as well as to Basile Bruneau who was author of an early version of this document.

Thanks also to Marcus Keane, Mikael Abrahamsson, Ray Bellis, Zhen Cao, Tim Chown, Lorenzo Colitti, Michael Di Bartolomeo, Ian Farrer,

Phillip Hallam-Baker, Bob Hinden, Tatuya Jinmei, Erik Kline, Ted Lemon, Paul Hoffman, Dave Thaler, Suresh Krishnan, Gorry Fairhurst, Jen Lenkova, Veronika McKillop, Mark Townsley and James Woodyatt for useful and interesting discussions and reviews.

Finally, special thanks to Thierry Danis for his valuable inputs and implementation efforts, Tom Jones for his integration effort into the NEAT project and Rigil Salim for his implementation work.

## 10. References

### 10.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [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>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <<https://www.rfc-editor.org/info/rfc3646>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4343] Eastlake 3rd, D., "Domain Name System (DNS) Case Insensitivity Clarification", RFC 4343, DOI 10.17487/RFC4343, January 2006, <<https://www.rfc-editor.org/info/rfc4343>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.



- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <<https://www.rfc-editor.org/info/rfc8106>>.
- [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>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <<https://www.rfc-editor.org/info/rfc8615>>.

## 10.2. Informative References

- [I-D.kline-mif-mpvd-api-reqs]  
Kline, E., "Multiple Provisioning Domains API Requirements", draft-kline-mif-mpvd-api-reqs-00 (work in progress), November 2015.

- [I-D.stenberg-mif-mpvd-dns]  
Stenberg, M. and S. Barth, "Multiple Provisioning Domains using Domain Name System", draft-stenberg-mif-mpvd-dns-00 (work in progress), October 2015.
- [IEEE8021X]  
"IEEE Standards for Local and Metropolitan Area Networks, Port-based Network Access Control, IEEE Std", n.d..
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <<https://www.rfc-editor.org/info/rfc4389>>.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<https://www.rfc-editor.org/info/rfc6105>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <<https://www.rfc-editor.org/info/rfc6147>>.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <<https://www.rfc-editor.org/info/rfc6296>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.

- [RFC7278] Byrne, C., Drown, D., and A. Vizdal, "Extending an IPv6 /64 Prefix from a Third Generation Partnership Project (3GPP) Mobile Interface to a LAN Link", RFC 7278, DOI 10.17487/RFC7278, June 2014, <<https://www.rfc-editor.org/info/rfc7278>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<https://www.rfc-editor.org/info/rfc7556>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [URN] "URN Namespaces", n.d..

## Authors' Addresses

Pierre Pfister  
Cisco  
11 Rue Camille Desmoulins  
Issy-les-Moulineaux 92130  
France

Email: [ppfister@cisco.com](mailto:ppfister@cisco.com)

Eric Vyncke  
Cisco  
De Kleetlaan, 6  
Diegem 1831  
Belgium

Email: [evyncke@cisco.com](mailto:evyncke@cisco.com)

Tommy Pauly  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014  
United States of America

Email: [tpauly@apple.com](mailto:tpauly@apple.com)

David Schinazi  
Google LLC  
1600 Amphitheatre Parkway  
Mountain View, California 94043  
United States of America

Email: dschinazi.ietf@gmail.com

Wenqin Shao  
Cisco  
11 Rue Camille Desmoulins  
Issy-les-Moulineaux 92130  
France

Email: wenshao@cisco.com

SPRING  
Internet-Draft  
Intended status: Standards Track  
Expires: July 2, 2021

C. Filsfils, Ed.  
P. Camarillo, Ed.  
Cisco Systems, Inc.  
J. Leddy  
Individual Contributor  
D. Voyer  
Bell Canada  
S. Matsushima  
SoftBank  
Z. Li  
Huawei Technologies  
December 29, 2020

SRv6 Network Programming  
draft-ietf-spring-srv6-network-programming-28

Abstract

The SRv6 Network Programming framework enables a network operator or an application to specify a packet processing program by encoding a sequence of instructions in the IPv6 packet header.

Each instruction is implemented on one or several nodes in the network and identified by an SRv6 Segment Identifier in the packet.

This document defines the SRv6 Network Programming concept and specifies the base set of SRv6 behaviors that enables the creation of interoperable overlays with underlay optimization.

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 July 2, 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

1. Introduction . . . . .	4
2. Terminology . . . . .	4
2.1. Requirements Language . . . . .	5
3. SRv6 SID . . . . .	5
3.1. SID Format . . . . .	6
3.2. SID Allocation within an SR domain . . . . .	7
3.3. SID Reachability . . . . .	9
4. SR Endpoint Behaviors . . . . .	10
4.1. End: Endpoint . . . . .	12
4.1.1. Upper-Layer Header . . . . .	12
4.2. End.X: Layer-3 Cross-Connect . . . . .	13
4.3. End.T: Specific IPv6 Table Lookup . . . . .	14
4.4. End.DX6: Decapsulation and IPv6 Cross-Connect . . . . .	14
4.5. End.DX4: Decapsulation and IPv4 Cross-Connect . . . . .	15
4.6. End.DT6: Decapsulation and Specific IPv6 Table Lookup . . . . .	16
4.7. End.DT4: Decapsulation and Specific IPv4 Table Lookup . . . . .	17
4.8. End.DT46: Decapsulation and Specific IP Table Lookup . . . . .	18
4.9. End.DX2: Decapsulation and L2 Cross-Connect . . . . .	19
4.10. End.DX2V: Decapsulation and VLAN L2 Table Lookup . . . . .	20
4.11. End.DT2U: Decapsulation and Unicast MAC L2 Table Lookup . . . . .	21
4.12. End.DT2M: Decapsulation and L2 Table Flooding . . . . .	22
4.13. End.B6.Encaps: Endpoint Bound to an SRv6 Policy w/ Encaps . . . . .	22
4.14. End.B6.Encaps.Red: End.B6.Encaps with Reduced SRH . . . . .	24
4.15. End.BM: Endpoint Bound to an SR-MPLS Policy . . . . .	24
4.16. Flavors . . . . .	25
4.16.1. PSP: Penultimate Segment Pop of the SRH . . . . .	25
4.16.2. USP: Ultimate Segment Pop of the SRH . . . . .	28
4.16.3. USD: Ultimate Segment Decapsulation . . . . .	28
5. SR Policy Headend Behaviors . . . . .	29
5.1. H.Encaps: SR Headend with Encapsulation in an SRv6 Policy . . . . .	30
5.2. H.Encaps.Red: H.Encaps with Reduced Encapsulation . . . . .	31

5.3.	H.Encaps.L2: H.Encaps Applied to Received L2 Frames . . .	31
5.4.	H.Encaps.L2.Red: H.Encaps.Red Applied to Received L2 frames . . . . .	31
6.	Counters . . . . .	32
7.	Flow-based Hash Computation . . . . .	32
8.	Control Plane . . . . .	32
8.1.	IGP . . . . .	33
8.2.	BGP-LS . . . . .	33
8.3.	BGP IP/VPN/EVPN . . . . .	33
8.4.	Summary . . . . .	33
9.	Security Considerations . . . . .	35
10.	IANA Considerations . . . . .	35
10.1.	Ethernet Next Header Type . . . . .	35
10.2.	SRv6 Endpoint Behaviors Registry . . . . .	36
10.2.1.	Initial Registrations . . . . .	36
11.	Acknowledgements . . . . .	38
12.	Contributors . . . . .	38
13.	References . . . . .	41
13.1.	Normative References . . . . .	41
13.2.	Informative References . . . . .	42
	Authors' Addresses . . . . .	43

## 1. Introduction

Segment Routing [RFC8402] leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called segments. Each one of these instructions represents a function to be called at a specific location in the network. A function is locally defined on the node where it is executed and may range from simply moving forward in the Segment List to any complex user-defined behavior. Network programming combines segment routing functions, both simple and complex, to achieve a networking objective that goes beyond mere packet routing.

This document defines the SRv6 Network Programming concept and specifies the main segment routing behaviors to enable the creation of interoperable overlays with underlay optimization.

The companion document [I-D.filsfils-spring-srv6-net-pgm-illustration] illustrates the concepts defined in this document.

Familiarity with the Segment Routing Header [RFC8754] is expected.

## 2. Terminology

The following terms used within this document are defined in [RFC8402]: Segment Routing, SR Domain, Segment ID (SID), SRv6, SRv6 SID, SR Policy, Prefix-SID, and Adj-SID.

The following terms used within this document are defined in [RFC8754]: SRH, SR Source Node, Transit Node, SR Segment Endpoint Node, Reduced SRH, Segments Left and Last Entry.

SL: The Segments Left field of the SRH

FIB: Forwarding Information Base. A FIB lookup is a lookup in the forwarding table.

SA: Source Address

DA: Destination Address

SRv6 SID function: The function part of the SID is an opaque identification of a local behavior bound to the SID. It is formally defined in Section 3.1 of this document.

SRv6 Segment Endpoint behavior: A packet processing behavior executed at an SRv6 Segment Endpoint Node. Section 4 of this document defines SRv6 Segment Endpoint behaviors related to traffic-engineering and



overlay use-cases. Other behaviors (e.g. service programming) are outside the scope of this document.

An SR Policy is resolved to a SID list. A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.

(SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:

- Source Address is SA, Destination Address is DA, and next-header is SRH.
- SRH with SID list <S1, S2, S3> with Segments Left = SL.
- Note the difference between the <> and () symbols: <S1, S2, S3> represents a SID list where S1 is the first SID and S3 is the last SID to traverse. (S3, S2, S1; SL) represents the same SID list but encoded in the SRH format where the rightmost SID in the SRH is the first SID and the leftmost SID in the SRH is the last SID. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of the detailed packet behavior, the (S3, S2, S1; SL) notation is more convenient.
- The payload of the packet is omitted.

Per-VRF VPN label: a single label for the entire VRF that is shared by all routes from that VRF ([RFC4364] Section 4.3.2)

Per-CE VPN label: a single label for each attachment circuit that is shared by all routes with the same "outgoing attachment circuit" ([RFC4364] Section 4.3.2)

## 2.1. Requirements Language

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. SRv6 SID

RFC8402 defines an SRv6 Segment Identifier as an IPv6 address explicitly associated with the segment.

When an SRv6 SID is in the Destination Address field of an IPv6 header of a packet, it is routed through Transit Nodes in an IPv6 network as an IPv6 address.

Its processing is defined in [RFC8754] section 4.3 and reproduced here as a reminder.

Without constraining the details of an implementation, the SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.

When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packet's destination address. This lookup can return any of the following:

- \* A FIB entry that represents a locally instantiated SRv6 SID
- \* A FIB entry that represents a local interface, not locally instantiated as an SRv6 SID
- \* A FIB entry that represents a nonlocal route
- \* No Match

Section 4 of this document defines a new set of SRv6 SID behaviors in addition to that defined in [RFC8754] Section 4.3.1.

### 3.1. SID Format

This document defines an SRv6 SID as consisting of LOC:FUNCT:ARG, where a locator (LOC) is encoded in the L most significant bits of the SID, followed by F bits of function (FUNCT) and A bits of arguments (ARG). L, the locator length, is flexible, and an operator is free to use the locator length of their choice. F and A may be any value as long as  $L+F+A \leq 128$ . When  $L+F+A$  is less than 128 then the remaining bits of the SID MUST be zero.

A locator may be represented as B:N where B is the SRv6 SID block (IPv6 prefix allocated for SRv6 SIDs by the operator) and N is the identifier of the parent node instantiating the SID.

When the LOC part of the SRv6 SIDs is routable, it leads to the node which instantiates the SID.

The FUNCT is an opaque identification of a local behavior bound to the SID.

The term "function" refers to the bit-string in the SRv6 SID. The term "behavior" identifies the behavior bound to the SID. Some behaviors are defined in Section 4 of this document.

An SRv6 Segment Endpoint Behavior may require additional information for its processing (e.g. related to the flow or service). This information may be encoded in the ARG bits of the SID.

In such a case, the semantics and format of the ARG bits are defined as part of the SRv6 endpoint behavior specification.

The ARG value of a routed SID SHOULD remain constant among packets in a given flow. Varying ARG values among packets in a flow may result in different ECMP hashing and cause re-ordering.

### 3.2. SID Allocation within an SR domain

Locators are assigned consistent with IPv6 infrastructure allocation. For example, a network operator may:

- o Assign block B::/48 to the SR domain
- o Assign a unique B:N::/64 block to each SRv6-enabled node in the domain

As an example, one mobile service provider has commercially deployed SRv6 across more than 1000 commercial routers and 1800 whitebox routers. All these devices are enabled for SRv6 and advertise SRv6 SIDs. The provider historically deployed IPv6 and assigned infrastructure addresses from ULA space [RFC4193]. They specifically allocated three /48 prefixes (Country X, Country Y, Country Z) to support their SRv6 infrastructure. From those /48 prefixes each router was assigned a /64 prefix from which all SIDs of that router are allocated.

In another example, a large mobile and fixed-line service provider has commercially deployed SRv6 in their country-wide network. This provider is assigned a /20 prefix by an RIR (Regional Internet Registry). They sub-allocated a few /48 prefixes to their infrastructure to deploy SRv6. Each router is assigned a /64 prefix from which all SIDs of that router are allocated.

IPv6 address consumption in both these examples is minimal, representing less than one billionth and one millionth of the available address space, respectively.

A service provider receiving the current minimum allocation of a /32 from an RIR may assign a /48 prefix to their infrastructure deploying

SRv6, and subsequently allocate /64 prefixes for SIDs at each SRv6 node. The /48 assignment is one sixty-five thousandth ( $1/2^{16}$ ) of the usable IPv6 address space available for assignment by the provider.

When an operator instantiates a SID at a node, they specify a SID value B:N:FUNCT and the behavior bound to the SID using one of the SRv6 Endpoint Behavior codepoint of the registry defined in this document (see Table 4).

The node advertises the SID, B:N:FUNCT, in the control-plane (see Section 8) together with the SRv6 Endpoint Behavior codepoint identifying the behavior of the SID.

An SR Source Node cannot infer the behavior by examination of the FUNCT value of a SID.

Therefore, the SRv6 Endpoint Behavior codepoint is advertised along with the SID in the control plane.

An SR Source Node uses the SRv6 Endpoint Behavior codepoint to map the received SID (B:N:FUNCT) to a behavior.

An SR Source Node selects a desired behavior at an advertising node by selecting the SID (B:N:FUNCT) advertised with the desired behavior.

As an example, a network operator may:

- o Assign an SRv6 SID block 2001:db8:bbbb::/48 from their in-house operation block for their SRv6 infrastructure
- o Assign an SRv6 Locator 2001:db8:bbbb:3::/64 to one particular router, for example Router 3, in their SR Domain
- o At Router 3, within the locator 2001:db8:bbbb:3::/64, the network operator or the router performs dynamic assignment for:
  - \* Function 0x0100 associated with the behavior End.X (Endpoint with cross-connect) between router 3 and its connected neighbor router, for example Router 4. This function is encoded as 16-bit value and has no arguments (F=16, A=0). This SID is advertised in the control plane as 2001:db8:bbbb:3:100:: with SRv6 Endpoint Behavior codepoint value of 5.
  - \* Function 0x0101 associated with the behavior End.X (Endpoint with cross-connect) between router 3 and its connected neighbor

router, for example Router 2. This function is encoded as 16-bit value and has no arguments (F=16, A=0). This SID is advertised in the control plane as 2001:db8:bbbb:3:101:: with SRv6 Endpoint Behavior codepoint value of 5.

These examples do not preclude any other IPv6 addressing allocation scheme.

### 3.3. SID Reachability

Most often, the node N would advertise IPv6 prefix(es) matching the LOC parts covering its SIDs or shorter-mask prefix. The distribution of these advertisements and calculation of their reachability are specific to the routing protocol and are outside of the scope of this document.

An SRv6 SID is said to be routed if its SID belongs to an IPv6 prefix advertised via a routing protocol. An SRv6 SID that does not fulfill this condition is non-routed.

Let's provide a classic illustration:

Node N is configured explicitly with two SIDs: 2001:db8:b:1:100:: and 2001:db8:b:2:101::.

The network learns about a path to 2001:db8:b:1::/64 via the IGP and hence a packet destined to 2001:db8:b:1:100:: would be routed up to N. The network does not learn about a path to 2001:db8:b:2::/64 via the IGP and hence a packet destined to 2001:db8:b:2:101:: would not be routed up to N.

A packet could be steered through a non-routed SID 2001:db8:b:2:101:: by using a SID list <...,2001:db8:b:1:100::,2001:db8:b:2:101::,...> where the non-routed SID is preceded by a routed SID to the same node. A packet could also be steered to a node instantiating a non-routed SID by preceding it in the SID-list with an Adjacency SID to that node. Routed and non-routed SRv6 SIDs are the SRv6 instantiation of global and local segments, respectively [RFC8402].

#### 4. SR Endpoint Behaviors

Following is a set of well-known behaviors that can be associated with a SID.

End	Endpoint function
End.X	The SRv6 instantiation of a Prefix SID [RFC8402]
End.T	Endpoint with Layer-3 cross-connect
End.DX6	The SRv6 instantiation of an Adj SID [RFC8402]
End.DX4	Endpoint with specific IPv6 table lookup
End.DT6	Endpoint with decapsulation and IPv6 cross-connect
End.DT4	e.g. IPv6-L3VPN (equivalent to per-CE VPN label)
End.DT46	Endpoint with decaps and IPv4 cross-connect
End.DX2	e.g. IPv4-L3VPN (equivalent to per-CE VPN label)
End.DX2V	Endpoint with decapsulation and IPv6 table lookup
End.DT2U	e.g. IPv6-L3VPN (equivalent to per-VRF VPN label)
End.DT2M	Endpoint with decapsulation and IPv4 table lookup
End.B6.Encaps	e.g. IPv4-L3VPN (equivalent to per-VRF VPN label)
End.B6.Encaps.Red	Endpoint with decapsulation and IP table lookup
End.BM	e.g. IP-L3VPN (equivalent to per-VRF VPN label)
	Endpoint with decapsulation and L2 cross-connect
	e.g. L2VPN use-case
	Endpoint with decaps and VLAN L2 table lookup
	e.g. EVPN Flexible cross-connect use-case
	Endpoint with decaps and unicast MAC L2 table lookup
	e.g. EVPN Bridging unicast use-case
	Endpoint with decapsulation and L2 table flooding
	e.g. EVPN Bridging BUM use-case with ESI filtering
	Endpoint bound to an SRv6 policy with encapsulation
	SRv6 instantiation of a Binding SID
	End.B6.Encaps with reduced SRH
	SRv6 instantiation of a Binding SID
	Endpoint bound to an SR-MPLS Policy
	SRv6 instantiation of an SR-MPLS Binding SID

The list is not exhaustive. In practice, any behavior can be attached to a local SID: e.g. a node N can bind a SID to a local VM or container which can apply any complex processing on the packet, provided there is a behavior codepoint allocated for the processing.

When an SRv6-capable node (N) receives an IPv6 packet whose destination address matches a FIB entry that represents a locally instantiated SRv6 SID (S), the IPv6 header chain is processed as defined in Section 4 of [RFC8200]. For SRv6 SIDs associated with an Endpoint Behavior defined in this document, the SRH and Upper-layer Header are processed as defined in the following subsections.

The pseudocode describing these behaviors details local processing at a node. An implementation of the pseudocode is compliant as long as the externally observable wire protocol is as described by the pseudocode.

Section 4.16 defines flavors of some of these behaviors.

Section 10.2 of this document defines the IANA Registry used to maintain all these behaviors as well as future ones defined in other documents.

#### 4.1. End: Endpoint

The Endpoint behavior ("End" for short) is the most basic behavior. It is the instantiation of a Prefix-SID [RFC8402].

When N receives a packet whose IPv6 DA is S and S is a local End SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.     Stop processing the SRH, and proceed to process the next
        header in the packet, whose type is identified by
        the Next Header field in the routing header.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
        Code 0 (Hop limit exceeded in transit),
        interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > Last Entry+1)) {
S10.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S11.   }
S12.   Decrement IPv6 Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Update IPv6 DA with Segment List[Segments Left]
S15.   Submit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
S16. }
```

Notes:

The End behavior operates on the same FIB table (i.e. identified by VRF or L3 relay id) associated to the packet. Hence the FIB lookup on line S15 is done in the same FIB table as the ingress interface.

##### 4.1.1. Upper-Layer Header

When processing the Upper-layer Header of a packet matching a FIB entry locally instantiated as an End SID, N does:



```
S01. If (Upper-Layer Header type is allowed by local configuration) {
S02.   Proceed to process the Upper-layer Header
S03. } Else {
S04.   Send an ICMP Parameter Problem to the Source Address,
       Code 4 (SR Upper-layer Header Error),
       Pointer set to the offset of the Upper-layer Header,
       Interrupt packet processing and discard the packet.
S05 }
```

Allowing processing of specific Upper-Layer Headers types is useful for OAM. As an example, an operator might permit pinging of SIDs. To do this they may enable local configuration to allow Upper-layer Header type 58 (ICMPv6).

It is RECOMMENDED that an implementation of local configuration only allows Upper-layer Header processing of types that do not result in the packet being forwarded (e.g. ICMPv6).

#### 4.2. End.X: Layer-3 Cross-Connect

The "Endpoint with cross-connect to an array of layer-3 adjacencies" behavior (End.X for short) is a variant of the End behavior.

It is the SRv6 instantiation of an Adjacency-SID [RFC8402] and its main use is for traffic-engineering policies.

Any SID instance of this behavior is associated with a set, J, of one or more Layer-3 adjacencies.

When N receives a packet destined to S and S is a local End.X SID, the line S15 from the End processing is replaced by the following:

```
S15.   Submit the packet to the IPv6 module for transmission
       to the new destination via a member of J
```

##### Notes:

S15. If the set J contains several L3 adjacencies, then one element of the set is selected based on a hash of the packet's header (see Section 7).

If a node N has 30 outgoing interfaces to 30 neighbors, usually the operator would explicitly instantiate 30 End.X SIDs at N: one per layer-3 adjacency to a neighbor. Potentially, more End.X could be explicitly defined (groups of layer-3 adjacencies to the same neighbor or to different neighbors).

Note that if N has an outgoing interface bundle I to a neighbor Q made of 10 member links, N might allocate up to 11 End.X local SIDs: one for the bundle itself and then up to one for each Layer-2 member link. The flows steered using the End.X SID corresponding to the bundle itself get load balanced across the member links via hashing while the flows steered using the End.X SID corresponding to a member link get steered over that specific member link alone.

When the End.X behavior is associated with a BGP Next-Hop, it is the SRv6 instantiation of the BGP Peering Segments [RFC8402].

When processing the Upper-layer Header of a packet matching a FIB entry locally instantiated as an End.X SID, process the packet as per Section 4.1.1.

#### 4.3. End.T: Specific IPv6 Table Lookup

The "Endpoint with specific IPv6 table lookup" behavior (End.T for short) is a variant of the End behavior.

The End.T behavior is used for multi-table operation in the core. For this reason, an instance of the End.T behavior is associated with an IPv6 FIB table T.

When N receives a packet destined to S and S is a local End.T SID, the line S15 from the End processing is replaced by the following:

- S15.1. Set the packet's associated FIB table to T
- S15.2. Submit the packet to the egress IPv6 FIB lookup and transmission to the new destination

When processing the Upper-layer Header of a packet matching a FIB entry locally instantiated as an End.T SID, process the packet as per Section 4.1.1.

#### 4.4. End.DX6: Decapsulation and IPv6 Cross-Connect

The "Endpoint with decapsulation and cross-connect to an array of IPv6 adjacencies" behavior (End.DX6 for short) is a variant of the End.X behavior.

One of the applications of the End.DX6 behavior is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress

Provider Edge (PE) is not required. This is equivalent to the per-CE VPN label in MPLS [RFC4364].

The End.DX6 SID MUST be the last segment in a SR Policy, and it is associated with one or more L3 IPv6 adjacencies J.

When N receives a packet destined to S and S is a local End.DX6 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DX6 SID, N does:

```
S01. If (Upper-Layer Header type == 41(IPv6) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Forward the exposed IPv6 packet to the L3 adjacency J
S04. } Else {
S05.   Process as per Section 4.1.1
S06. }
```

Notes:

S01. 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

S03. If the End.DX6 SID is bound to an array of L3 adjacencies, then one entry of the array is selected based on the hash of the packet's header (see Section 7).

#### 4.5. End.DX4: Decapsulation and IPv4 Cross-Connect

The "Endpoint with decapsulation and cross-connect to an array of IPv4 adjacencies" behavior (End.DX4 for short) is a variant of the End.X behavior.

One of the applications of the End.DX4 behavior is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This is equivalent to the per-CE VPN label in MPLS [RFC4364].

The End.DX4 SID MUST be the last segment in a SR Policy, and it is associated with one or more L3 IPv4 adjacencies J.

When N receives a packet destined to S and S is a local End.DX4 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DX4 SID, N does:

```
S01. If (Upper-Layer Header type == 4(IPv4) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Forward the exposed IPv4 packet to the L3 adjacency J
S04. } Else {
S05.   Process as per Section 4.1.1
S06. }
```

Notes:

S01. 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers

S03. If the End.DX4 SID is bound to an array of L3 adjacencies, then one entry of the array is selected based on the hash of the packet's header (see Section 7).

#### 4.6. End.DT6: Decapsulation and Specific IPv6 Table Lookup

The "Endpoint with decapsulation and specific IPv6 table lookup" behavior (End.DT6 for short) is a variant of the End.T behavior.

One of the applications of the End.DT6 behavior is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This is equivalent to the per-VRF VPN label in MPLS [RFC4364].

Note that an End.DT6 may be defined for the main IPv6 table in which case an End.DT6 supports the equivalent of an IPv6inIPv6 decapsulation (without VPN/tenant implication).

The End.DT6 SID MUST be the last segment in a SR Policy, and a SID instance is associated with an IPv6 FIB table T.

When N receives a packet destined to S and S is a local End.DT6 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DT6 SID, N does:

```
S01. If (Upper-Layer Header type == 41(IPv6) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Set the packet's associated FIB table to T
S04.   Submit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
S05. } Else {
S06.   Process as per Section 4.1.1
S07. }
```

#### 4.7. End.DT4: Decapsulation and Specific IPv4 Table Lookup

The "Endpoint with decapsulation and specific IPv4 table lookup" behavior (End.DT4 for short) is a variant of the End.T behavior.

One of the applications of the End.DT4 behavior is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This is equivalent to the per-VRF VPN label in MPLS [RFC4364].

Note that an End.DT4 may be defined for the main IPv4 table in which case an End.DT4 supports the equivalent of an IPv4inIPv6 decapsulation (without VPN/tenant implication).

The End.DT4 SID MUST be the last segment in a SR Policy, and a SID instance is associated with an IPv4 FIB table T.

When N receives a packet destined to S and S is a local End.DT4 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DT4 SID, N does:

```
S01. If (Upper-Layer Header type == 4(IPv4) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Set the packet's associated FIB table to T
S04.   Submit the packet to the egress IPv4 FIB lookup and
        transmission to the new destination
S05. } Else {
S06.   Process as per Section 4.1.1
S07. }
```

#### 4.8. End.DT46: Decapsulation and Specific IP Table Lookup

The "Endpoint with decapsulation and specific IP table lookup" behavior (End.DT46 for short) is a variant of the End.DT4 and End.DT6 behavior.

One of the applications of the End.DT46 behavior is the L3VPN use-case where a FIB lookup in a specific IP tenant table at the egress PE is required. This is equivalent to single per-VRF VPN label (for IPv4 and IPv6) in MPLS[RFC4364].

Note that an End.DT46 may be defined for the main IP table in which case an End.DT46 supports the equivalent of an IPinIPv6 decapsulation(without VPN/tenant implication).

The End.DT46 SID MUST be the last segment in a SR Policy, and a SID instance is associated with an IPv4 FIB table T4 and an IPv6 FIB table T6.

When N receives a packet destined to S and S is a local End.DT46 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DT46 SID, N does:

```
S01. If (Upper-layer Header type == 4(IPv4) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Set the packet's associated FIB table to T4
S04.   Submit the packet to the egress IPv4 FIB lookup and
        transmission to the new destination
S05. } Else if (Upper-layer Header type == 41(IPv6) ) {
S06.   Remove the outer IPv6 Header with all its extension headers
S07.   Set the packet's associated FIB table to T6
S08.   Submit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
S09. } Else {
S10.   Process as per Section 4.1.1
S11. }
```

#### 4.9. End.DX2: Decapsulation and L2 Cross-Connect

The "Endpoint with decapsulation and Layer-2 cross-connect to an outgoing L2 interface (OIF)" (End.DX2 for short) is a variant of the endpoint behavior.

One of the applications of the End.DX2 behavior is the L2VPN [RFC4664] / EVPN VPWS [RFC7432] [RFC8214] use-case.

The End.DX2 SID MUST be the last segment in a SR Policy, and it is associated with one outgoing interface I.

When N receives a packet destined to S and S is a local End.DX2 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.DX2 SID, N does:

```
S01. If (Upper-Layer Header type == 143(Ethernet) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Forward the Ethernet frame to the OIF I
S04. } Else {
S05.   Process as per Section 4.1.1
S06. }
```

Notes:

S01. IANA has allocated the Internet Protocol number 143 to Ethernet [IEEE.802.3\_2018] (see Section 10.1).  
S03. An End.DX2 behavior could be customized to expect a specific IEEE header (e.g. VLAN tag) and rewrite the egress IEEE header before forwarding on the outgoing interface.

Note that an End.DX2 SID may also be associated with a bundle of outgoing interfaces.

#### 4.10. End.DX2V: Decapsulation and VLAN L2 Table Lookup

The "Endpoint with decapsulation and specific VLAN table lookup" behavior (End.DX2V for short) is a variant of the End.DX2 behavior.

One of the applications of the End.DX2V behavior is the EVPN Flexible cross-connect use-case. The End.DX2V behavior is used to perform a lookup of the Ethernet frame VLANs in a particular L2 table. Any SID instance of this behavior is associated with an L2 Table T.

When N receives a packet whose IPv6 DA is S and S is a local End.DX2 SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is modified as follows:



S03. Lookup the exposed VLANs in L2 table T, and forward via the matched table entry.

Notes:

S03. An End.DX2V behavior could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

#### 4.11. End.DT2U: Decapsulation and Unicast MAC L2 Table Lookup

The "Endpoint with decapsulation and specific unicast MAC L2 table lookup" behavior (End.DT2U for short) is a variant of the End behavior.

One of the applications of the End.DT2U behavior is the EVPN Bridging unicast [RFC7432]. Any SID instance of the End.DT2U behavior is associated with an L2 Table T.

When N receives a packet whose IPv6 DA is S and S is a local End.DT2U SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is as follows:

```
S01. If (Upper-Layer Header type == 143(Ethernet) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Learn the exposed MAC Source Address in L2 Table T
S04.   Lookup the exposed MAC Destination Address in L2 Table T
S05.   If (matched entry in T) {
S06.     Forward via the matched table T entry
S07.   } Else {
S08.     Forward via all L2 OIFs entries in table T
S09.   }
S10. } Else {
S11.   Process as per Section 4.1.1
S12. }
```

Notes:

S01. IANA has allocated the Internet Protocol number 143 to Ethernet (see Section 10.1).

S03. In EVPN [RFC7432], the learning of the exposed MAC Source Address is done via control plane. In L2VPN VPLS [RFC4761] [RFC4762] reachability is obtained by standard learning bridge functions in the data plane.

#### 4.12. End.DT2M: Decapsulation and L2 Table Flooding

The "Endpoint with decapsulation and specific L2 table flooding" behavior (End.DT2M for short) is a variant of the End.DT2U behavior.

Two of the applications of the End.DT2M behavior are the EVPN Bridging of broadcast, unknown and multicast (BUM) traffic with Ethernet Segment Identifier (ESI) filtering [RFC7432] and the EVPN ETREE [RFC8317] use-cases.

Any SID instance of this behavior is associated with a L2 table T. The behavior also takes an argument: "Arg.FE2". This argument provides a local mapping to ESI for split-horizon filtering of the received traffic to exclude specific OIF (or set of OIFs) from L2 table T flooding. The allocation of the argument values is local to the SR Endpoint Node instantiating this behavior and the signaling of the argument to other nodes for the EVPN functionality occurs via control plane.

When N receives a packet whose IPv6 DA is S and S is a local End.DT2M SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is as follows:

```
S01. If (Upper-Layer Header type == 143(Ethernet) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Learn the exposed MAC Source Address in L2 Table T
S04.   Forward via all L2OIFs excluding those associated by the
       identifier Arg.FE2
S05. } Else {
S06.   Process as per Section 4.1.1
S07. }
```

Notes:

S01. IANA has allocated the Internet Protocol number 143 to Ethernet (see Section 10.1).

S03. In EVPN [RFC7432], the learning of the exposed MAC Source Address is done via control plane. In L2VPN VPLS [RFC4761] [RFC4762] reachability is obtained by standard learning bridge functions in the data plane.

#### 4.13. End.B6.Encaps: Endpoint Bound to an SRv6 Policy w/ Encaps

This is a variation of the End behavior.

One of its applications is to express scalable traffic-engineering policies across multiple domains. It is one of the SRv6 instantiations of a Binding SID [RFC8402].

Any SID instance of this behavior is associated with an SR Policy B and a source address A.

When N receives a packet whose IPv6 DA is S and S is a local End.B6.Encaps SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.     Stop processing the SRH, and proceed to process the next
        header in the packet, whose type is identified by
        the Next Header field in the routing header.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
        Code 0 (Hop limit exceeded in transit),
        interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > (Last Entry+1))) {
S10.     Send an ICMP Parameter Problem to the Source Address,
        Code 0 (Erroneous header field encountered),
        Pointer set to the Segments Left field,
        interrupt packet processing and discard the packet.
S11.   }
S12.   Decrement IPv6 Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Update IPv6 DA with Segment List[Segments Left]
S15.   Push a new IPv6 header with its own SRH containing B
S16.   Set the outer IPv6 SA to A
S17.   Set the outer IPv6 DA to the first SID of B
S18.   Set the outer Payload Length, Traffic Class, Flow Label,
        Hop Limit and Next-Header fields
S19.   Submit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
S20. }
```

Notes:

S15. The SRH MAY be omitted when the SRv6 Policy B only contains one SID and there is no need to use any flag, tag or TLV.

S18. The Payload Length, Traffic Class, Hop Limit and Next-Header fields are set as per [RFC2473]. The Flow Label is computed as per [RFC6437].

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.B6.Encaps SID, process the packet as per Section 4.1.1.

#### 4.14. End.B6.Encaps.Red: End.B6.Encaps with Reduced SRH

This is an optimization of the End.B6.Encaps behavior.

End.B6.Encaps.Red reduces the size of the SRH by one SID by excluding the first SID in the SRH of the new IPv6 header. Thus, the first segment is only placed in the IPv6 Destination Address of the new IPv6 header and the packet is forwarded according to it.

The SRH Last Entry field is set as defined in Section 4.1.1 of [RFC8754].

The SRH MAY be omitted when the SRv6 Policy only contains one SID and there is no need to use any flag, tag or TLV.

#### 4.15. End.BM: Endpoint Bound to an SR-MPLS Policy

The "Endpoint bound to an SR-MPLS Policy" is a variant of the End behavior.

The End.BM behavior is required to express scalable traffic-engineering policies across multiple domains where some domains support the MPLS instantiation of Segment Routing. This is an SRv6 instantiation of an SR-MPLS Binding SID [RFC8402].

Any SID instance of this behavior is associated with an SR-MPLS Policy B.

When N receives a packet whose IPv6 DA is S and S is a local End.BM SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.     Stop processing the SRH, and proceed to process the next
           header in the packet, whose type is identified by
           the Next Header field in the routing header.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.     Send an ICMP Time Exceeded message to the Source Address,
           Code 0 (Hop limit exceeded in transit),
           interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > (Last Entry+1))) {
S10.     Send an ICMP Parameter Problem to the Source Address,
           Code 0 (Erroneous header field encountered),
           Pointer set to the Segments Left field,
           interrupt packet processing and discard the packet.

S11.   }
S12.   Decrement IPv6 Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Update IPv6 DA with Segment List[Segments Left]
S15.   Push the MPLS label stack for B
S16.   Submit the packet to the MPLS engine for transmission
S17. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an End.BM SID, process the packet as per Section 4.1.1.

#### 4.16. Flavors

The Penultimate Segment Pop of the SRH (PSP), Ultimate Segment Pop of the SRH (USP) and Ultimate Segment Decapsulation (USD) flavors are variants of the End, End.X and End.T behaviors. The End, End.X and End.T behaviors can support these flavors either individually or in combinations.

##### 4.16.1. PSP: Penultimate Segment Pop of the SRH

###### 4.16.1.1. Guidelines

SR Segment Endpoint Nodes advertise the SIDs instantiated on them via control plane protocols as described in Section 8. Different behavior ids are allocated for flavored and unflavored SIDs (see Table 4).

An SR Segment Endpoint Node that offers both PSP and non-PSP flavored behavior advertises them as two different SIDs.

The SR Segment Endpoint Node only advertises the PSP flavor if the operator enables this capability at the node.

The PSP operation is deterministically controlled by the SR Source Node.

A PSP-flavored SID is used by the Source SR Node when it needs to instruct the penultimate SR Segment Endpoint Node listed in the SRH to remove the SRH from the IPv6 header.

#### 4.16.1.2. Definition

SR Segment Endpoint Nodes receive the IPv6 packet with the Destination Address field of the IPv6 Header equal to its SID address.

A penultimate SR Segment Endpoint Node is one that, as part of the SID processing, copies the last SID from the SRH into the IPv6 Destination Address and decrements the Segments Left value from one to zero.

The PSP operation only takes place at a penultimate SR Segment Endpoint Node and does not happen at any Transit Node. When a SID of PSP-flavor is processed at a non-penultimate SR Segment Endpoint Node, the PSP behavior is not performed as described in the pseudocode below since Segments Left would not be zero.

The SRH processing of the End, End.X and End.T behaviors are modified: after the instruction "S14. Update IPv6 DA with Segment List[Segments Left]" is executed, the following instructions must be executed as well:

```
S14.1.  If (Segments Left == 0) {
S14.2.      Update the Next Header field in the preceding header to the
           Next Header value from the SRH
S14.3.      Decrease the IPv6 header Payload Length by 8*(Hdr Ext Len+1)
S14.4.      Remove the SRH from the IPv6 extension header chain
S14.5.  }
```

The usage of PSP does not increase the MTU of the IPv6 packet and hence does not have any impact on the PMTU discovery mechanism.

As a reminder, [RFC8754] defines in section 5 the SR Deployment Model within the SR Domain [RFC8402]. Within this framework, the Authentication Header (AH) is not used to secure the SRH as described

in Section 7.5 of [RFC8754]. Hence, the discussion of applicability of PSP along with AH usage is beyond the scope of this document.

In the context of this specification, the End, End.X and End.T behaviors with PSP do not contravene Section 4 of [RFC8200] because the destination address of the incoming packet is the address of the node executing the behavior.

#### 4.16.1.3. Use-case

One use-case for the PSP functionality is streamlining the operation of an egress border router.

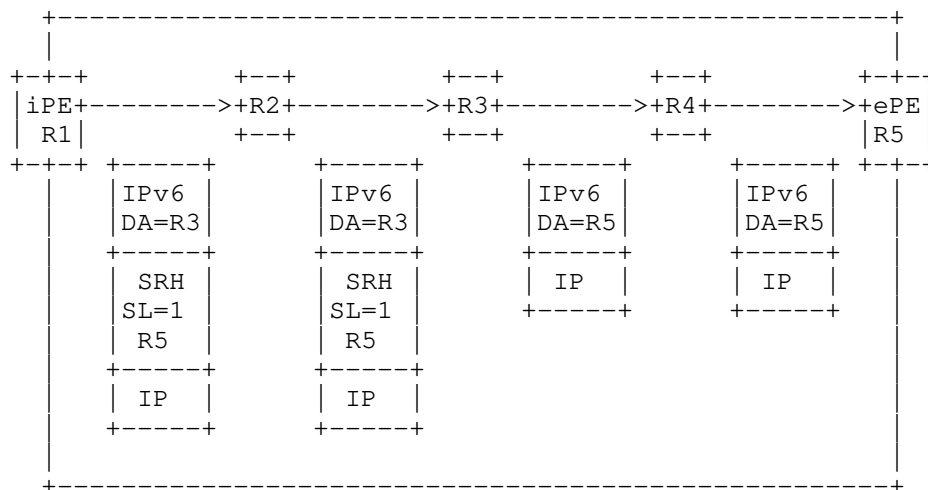


Figure 1: PSP use-case topology

In the above illustration, for a packet sent from iPE to ePE, node R3 is an intermediate traffic engineering waypoint and is the penultimate segment endpoint router; the node that copies the last segment from the SRH into the IPv6 Destination Address and decrements segments left to 0. The SDN controller knows that no other node after R3 needs to inspect the SRH, and it instructs R3 to remove the exhausted SRH from the packet by using a PSP-flavored SID.

The benefits for the egress PE are straightforward:

- as part of the decapsulation process the egress PE is required to parse and remove fewer bytes from the packet.
- if a lookup on an upper-layer IP header is required (e.g. per-VRF VPN), the header is more likely to be within the memory accessible

to the lookup engine in the forwarding ASIC (Application-specific integrated circuit).

#### 4.16.2. USP: Ultimate Segment Pop of the SRH

The SRH processing of the End, End.X and End.T behaviors are modified: the instructions S02-S04 are substituted by the following ones:

```
S02.    If (Segments Left == 0) {
S03.1.    Update the Next Header field in the preceding header to the
           Next Header value of the SRH
S03.2.    Decrease the IPv6 header Payload Length by 8*(Hdr Ext Len+1)
S03.3.    Remove the SRH from the IPv6 extension header chain
S03.4.    Proceed to process the next header in the packet
S04.    }
```

One of the applications of the USP flavor is when a packet with an SRH is destined to an application on hosts with smartNICs implementing SRv6. The USP flavor is used to remove the consumed SRH from the extension header chain before sending the packet to the host.

#### 4.16.3. USD: Ultimate Segment Decapsulation

The Upper-layer header processing of the End, End.X and End.T behaviors are modified as follows:

```
End:
S01. If (Upper-layer Header type == 41(IPv6) ) {
S02.    Remove the outer IPv6 Header with all its extension headers
S03.    Submit the packet to the egress IPv6 FIB lookup and
           transmission to the new destination
S04. } Else if (Upper-layer Header type == 4(IPv4) ) {
S05.    Remove the outer IPv6 Header with all its extension headers
S06.    Submit the packet to the egress IPv4 FIB lookup and
           transmission to the new destination
S07. Else {
S08.    Process as per Section 4.1.1
S09. }
```



```

End.T:
S01. If (Upper-layer Header type == 41(IPv6) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Set the packet's associated FIB table to T
S04.   Submit the packet to the egress IPv6 FIB lookup and
       transmission to the new destination
S05. } Else if (Upper-layer Header type == 4(IPv4) ) {
S06.   Remove the outer IPv6 Header with all its extension headers
S07.   Set the packet's associated FIB table to T
S08.   Submit the packet to the egress IPv4 FIB lookup and
       transmission to the new destination
S09. Else {
S10.   Process as per Section 4.1.1
S11. }

End.X:
S01. If (Upper-layer Header type == 41(IPv6) ||
       Upper-layer Header type == 4(IPv4) ) {
S02.   Remove the outer IPv6 Header with all its extension headers
S03.   Forward the exposed IP packet to the L3 adjacency J
S04. } Else {
S05.   Process as per Section 4.1.1
S06. }

```

One of the applications of the USD flavor is the case of TI-LFA in P routers with encapsulation. The USD flavor allows the last Segment Endpoint Node in the repair path list to decapsulate the IPv6 header added at the TI-LFA Point of Local Repair and forward the inner packet.

## 5. SR Policy Headend Behaviors

This section describes a set of SR Policy Headend [RFC8402] behaviors.

H.Encaps	SR Headend Behavior with Encapsulation in an SR Policy
H.Encaps.Red	H.Encaps with Reduced Encapsulation
H.Encaps.L2	H.Encaps Applied to Received L2 Frames
H.Encaps.L2.Red	H.Encaps.Red Applied to Received L2 Frames

This list is not exhaustive and future documents may define additional behaviors.

### 5.1. H.Encaps: SR Headend with Encapsulation in an SRv6 Policy

Node N receives two packets P1=(A, B2) and P2=(A,B2) (B3, B2, B1; SL=1). B2 is neither a local address nor SID of N.

Node N is configured with an IPv6 Address T (e.g. assigned to its loopback).

N steers the transit packets P1 and P2 into an SR Policy with a Source Address T and a Segment list <S1, S2, S3>.

The H.Encaps encapsulation behavior is defined as follows:

- S01. Push an IPv6 header with its own SRH
- S02. Set outer IPv6 SA = T and outer IPv6 DA to the first SID in the segment list
- S03. Set outer Payload Length, Traffic Class, Hop Limit and Flow Label fields
- S04. Set the outer Next-Header value
- S05. Decrement inner IPv6 Hop Limit or IPv4 TTL
- S06. Submit the packet to the IPv6 module for transmission to S1

Note:

S03: As described in [RFC2473] and [RFC6437].

After the H.Encaps behavior, P1' and P2' respectively look like:

- (T, S1) (S3, S2, S1; SL=2) (A, B2)
- (T, S1) (S3, S2, S1; SL=2) (A, B2) (B3, B2, B1; SL=1)

The received packet is encapsulated unmodified (with the exception of the IPv4 TTL or IPv6 Hop Limit that is decremented as described in [RFC2473]).

The H.Encaps behavior is valid for any kind of Layer-3 traffic. This behavior is commonly used for L3VPN with IPv4 and IPv6 deployments. It may be also used for TI-LFA [I-D.ietf-rtgwg-segment-routing-ti-lfa] at the point of local repair.

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

## 5.2. H.Encaps.Red: H.Encaps with Reduced Encapsulation

The H.Encaps.Red behavior is an optimization of the H.Encaps behavior.

H.Encaps.Red reduces the length of the SRH by excluding the first SID in the SRH of the pushed IPv6 header. The first SID is only placed in the Destination Address field of the pushed IPv6 header.

After the H.Encaps.Red behavior, P1' and P2' respectively look like:

- (T, S1) (S3, S2; SL=2) (A, B2)
- (T, S1) (S3, S2; SL=2) (A, B2) (B3, B2, B1; SL=1)

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

## 5.3. H.Encaps.L2: H.Encaps Applied to Received L2 Frames

The H.Encaps.L2 behavior encapsulates a received Ethernet [IEEE.802.3\_2018] frame and its attached VLAN header, if present, in an IPv6 packet with an SRH. The Ethernet frame becomes the payload of the new IPv6 packet.

The Next Header field of the SRH MUST be set to 143.

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

The encapsulating node MUST remove the preamble (if any) and frame check sequence (FCS) from the Ethernet frame upon encapsulation and the decapsulating node MUST regenerate, as required, the preamble and FCS before forwarding Ethernet frame.

## 5.4. H.Encaps.L2.Red: H.Encaps.Red Applied to Received L2 frames

The H.Encaps.L2.Red behavior is an optimization of the H.Encaps.L2 behavior.

H.Encaps.L2.Red reduces the length of the SRH by excluding the first SID in the SRH of the pushed IPv6 header. The first SID is only placed in the Destination Address field of the pushed IPv6 header.

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

## 6. Counters

A node supporting this document SHOULD implement a pair of traffic counters (one for packets and one for bytes) per local SID entry, for traffic that matched that SID and was processed successfully (i.e. packets which generate ICMP Error Messages or are dropped are not counted). The retrieval of these counters from MIB, NETCONF/YANG or any other data structure is outside the scope of this document.

## 7. Flow-based Hash Computation

When a flow-based selection within a set needs to be performed, the IPv6 Source Address, the IPv6 Destination Address and the IPv6 Flow Label of the outer IPv6 header MUST be included in the flow-based hash.

This occurs when a FIB lookup is performed and multiple ECMP paths exist to the updated destination address.

This occurs when End.X, End.DX4, or End.DX6 are bound to an array of adjacencies.

This occurs when the packet is steered in an SR policy whose selected path has multiple SID lists.

Additionally, any transit router in an SRv6 domain includes the outer flow label in its ECMP flow-based hash [RFC6437].

## 8. Control Plane

In an SDN environment, one expects the controller to explicitly provision the SIDs and/or discover them as part of a service discovery function. Applications residing on top of the controller could then discover the required SIDs and combine them to form a distributed network program.

The concept of "SRv6 network programming" refers to the capability for an application to encode any complex program as a set of individual functions distributed through the network. Some functions relate to underlay SLA, others to overlay/tenant, others to complex applications residing in VM and containers.

While not necessary for an SDN control plane, the remainder of this section provides a high-level illustrative overview of how control-plane protocols may be involved with SRv6. Their specification is outside the scope of this document.

### 8.1. IGP

The End, End.T and End.X SIDs express topological behaviors and hence are expected to be signaled in the IGP together with the flavors PSP, USP and USD. The IGP should also advertise the maximum SRv6 SID depth (MSD) capability of the node for each type of SRv6 operation - in particular, the SR source (e.g. H.Encaps), intermediate endpoint (e.g. End, End.X) and final endpoint (e.g. End.DX4, End.DT6) behaviors. These capabilities are factored in by an SR Source Node (or a controller) during the SR Policy computation.

The presence of SIDs in the IGP does not imply any routing semantics to the addresses represented by these SIDs. The routing reachability to an IPv6 address is solely governed by the non-SID-related IGP prefix reachability information that includes locators. Routing is neither governed nor influenced in any way by a SID advertisement in the IGP.

These SIDs provide important topological behaviors for the IGP to build FRR solutions based on TI-LFA [I-D.ietf-rtgwg-segment-routing-ti-lfa] and for TE processes relying on IGP topology database to build SR policies.

### 8.2. BGP-LS

BGP-LS provides the functionality for topology discovery that includes the SRv6 capabilities of the nodes, their locators and locally instantiated SIDs. This enables controllers or applications to build an inter-domain topology that can be used for computation of SR Policies using the SRv6 SIDs.

### 8.3. BGP IP/VPN/EVPN

The End.DX4, End.DX6, End.DT4, End.DT6, End.DT46, End.DX2, End.DX2V, End.DT2U and End.DT2M SIDs can be signaled in BGP.

In some scenarios an egress PE advertising a VPN route might wish to abstract the specific behavior bound to the SID from the ingress PE and other routers in the network. In such case, the SID may be advertised using the Opaque SRv6 Endpoint Behavior codepoint defined in Table 4. The details of such control plane signaling mechanisms are out of the scope of this document.

### 8.4. Summary

The following table summarizes behaviors for SIDs that can be signaled in which each respective control plane protocol.

	IGP	BGP-LS	BGP IP/VPN/EVPN
End (PSP, USP, USD)	X	X	
End.X (PSP, USP, USD)	X	X	
End.T (PSP, USP, USD)	X	X	
End.DX6	X	X	X
End.DX4	X	X	X
End.DT6	X	X	X
End.DT4	X	X	X
End.DT46	X	X	X
End.DX2		X	X
End.DX2V		X	X
End.DT2U		X	X
End.DT2M		X	X
End.B6.Encaps		X	
End.B6.Encaps.Red		X	
End.B6.BM		X	

Table 1: SRv6 locally instantiated SIDs signaling

The following table summarizes which SR Policy Headend capabilities are signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN/EVPN
H.Encaps	X	X	
H.Encaps.Red	X	X	
H.Encaps.L2		X	
H.Encaps.L2.Red		X	

Table 2: SRv6 Policy Headend behaviors signaling

The previous table describes generic capabilities. It does not describe specific instantiated SR policies.

For example, a BGP-LS advertisement of H.Encaps behavior would describe the capability of node N to perform a H.Encaps behavior. Specifically, it would describe how many SIDs could be pushed by N without significant performance degradation.

As a reminder, an SR policy is always assigned a Binding SID [RFC8402]. BSIDs are also advertised in BGP-LS as shown in Table 1.

Hence, the Table 2 only focuses on the generic capabilities related to H.Encaps.

## 9. Security Considerations

The security considerations for Segment Routing are discussed in [RFC8402]. Section 5 of [RFC8754] describes the SR Deployment Model and the requirements for securing the SR Domain. The security considerations of [RFC8754] also cover topics such as attack vectors and their mitigation mechanisms that also apply the behaviors introduced in this document. Together, they describe the required security mechanisms that allow establishment of an SR domain of trust. Having such a well-defined trust boundary is necessary in order to operate SRv6-based services for internal traffic while preventing any external traffic from accessing or exploiting the SRv6-based services. Care and rigor in IPv6 address allocation for use for SRv6 SID allocations and network infrastructure addresses, as distinct from IPv6 addresses allocated for end-users/systems (as illustrated in Section 5.1 of [RFC8754]), can provide the clear distinction between internal and external address space that is required to maintain the integrity and security of the SRv6 Domain. Additionally, [RFC8754] defines an HMAC TLV permitting SR Endpoint Nodes in the SR domain to verify that the SRH applied to a packet was selected by an authorized party and to ensure that the segment list is not modified after generation, regardless of the number of segments in the segment list. When enabled by local configuration, HMAC processing occurs at the beginning of SRH processing as defined in [RFC8754] Section 2.1.2.1 .

This document introduces SRv6 Endpoint and SR Policy Headend behaviors for implementation on SRv6 capable nodes in the network. The headend policy definition should be consistent with the specific behavior used and any local configuration (as specified in Section 4.1.1). As such, this document does not introduce any new security considerations.

The SID Behaviors specified in this document have the same HMAC TLV handling and mutability properties of the Flags, Tag, and Segment List field as the SID Behavior specified in [RFC8754].

## 10. IANA Considerations

### 10.1. Ethernet Next Header Type

This document requests IANA to allocate, in the "Protocol Numbers" registry (<https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>), a new value for "Ethernet" with the following definition: The value 143 in the Next Header field of an IPv6 header

or any extension header indicates that the payload is an Ethernet frame [IEEE.802.3\_2018].

IANA has done a temporary allocation of Protocol Number 143.

## 10.2. SRv6 Endpoint Behaviors Registry

This document requests IANA to create a new top-level registry called "Segment Routing Parameters". This registry is being defined to serve as a top-level registry for keeping all other Segment Routing sub-registries.

Additionally, a new sub-registry "SRv6 Endpoint Behaviors" is to be created under top-level "Segment Routing Parameters" registry. This sub-registry maintains 16-bit identifiers for the SRv6 Endpoint behaviors. This registry is established to provide consistency for control plane protocols which need to refer to these behaviors. These values are not encoded in the function bits within a SID.

The range of the registry is 0-65535 (0x0000 - 0xFFFF) and has the following registration rules and allocation policies:

Range	Hex	Registration procedure	Notes
0	0x0000	Reserved	Not to be allocated
1-32767	0x0001-0x7FFF	First Come First Served [RFC8126]	
32768-34815	0x8000-0x87FF	Private Use [RFC8126]	
34816-65534	0x8800-0xFFFFE	Reserved	Opaque
65535	0xFFFF	Reserved	

Table 3: SRv6 Endpoint Behaviors Registry

### 10.2.1. Initial Registrations

The initial registrations for the sub-registry are as follows:

Value	Hex	Endpoint behavior	Reference
0	0x0000	Reserved	Not to be allocated [This.ID]
1	0x0001	End	



2	0x0002	End with PSP	[This.ID]
3	0x0003	End with USP	[This.ID]
4	0x0004	End with PSP&USP	[This.ID]
5	0x0005	End.X	[This.ID]
6	0x0006	End.X with PSP	[This.ID]
7	0x0007	End.X with USP	[This.ID]
8	0x0008	End.X with PSP&USP	[This.ID]
9	0x0009	End.T	[This.ID]
10	0x000A	End.T with PSP	[This.ID]
11	0x000B	End.T with USP	[This.ID]
12	0x000C	End.T with PSP&USP	[This.ID]
14	0x000E	End.B6.Encaps	[This.ID]
15	0x000F	End.BM	[This.ID]
16	0x0010	End.DX6	[This.ID]
17	0x0011	End.DX4	[This.ID]
18	0x0012	End.DT6	[This.ID]
19	0x0013	End.DT4	[This.ID]
20	0x0014	End.DT46	[This.ID]
21	0x0015	End.DX2	[This.ID]
22	0x0016	End.DX2V	[This.ID]
23	0x0017	End.DT2U	[This.ID]
24	0x0018	End.DT2M	[This.ID]
25	0x0019	Reserved	[This.ID]
27	0x001B	End.B6.Encaps.Red	[This.ID]
28	0x001C	End with USD	[This.ID]
29	0x001D	End with PSP&USD	[This.ID]
30	0x001E	End with USP&USD	[This.ID]
31	0x001F	End with PSP, USP & USD	[This.ID]
32	0x0020	End.X with USD	[This.ID]
33	0x0021	End.X with PSP&USD	[This.ID]
34	0x0022	End.X with USP&USD	[This.ID]
35	0x0023	End.X with PSP, USP & USD	[This.ID]
36	0x0024	End.T with USD	[This.ID]
37	0x0025	End.T with PSP&USD	[This.ID]
38	0x0026	End.T with USP&USD	[This.ID]
39	0x0027	End.T with PSP, USP & USD	[This.ID]
40-32766		Unassigned	
32767	0x7FFF	The SID defined in RFC8754	[This.ID] [RFC8754]
32768-65534		Reserved	
65535	0xFFFF	Opaque	[This.ID]

Table 4: IETF - SRv6 Endpoint Behaviors

## 11. Acknowledgements

The authors would like to acknowledge Stefano Previdi, Dave Barach, Mark Townsley, Peter Psenak, Thierry Couture, Kris Michielsen, Paul Wells, Robert Hanzl, Dan Ye, Gaurav Dawra, Faisal Iqbal, Jaganbabu Rajamanickam, David Toscano, Asif Islam, Jianda Liu, Yunpeng Zhang, Jiaoming Li, Narendra A.K, Mike Mc Gourty, Bhupendra Yadav, Sherif Toulan, Satish Damodaran, John Bettink, Kishore Nandyala Veera Venk, Jisu Bhattacharya, Saleem Hafeez and Brian Carpenter.

## 12. Contributors

Daniel Bernier  
Bell Canada  
Canada

Email: daniel.bernier@bell.ca

Dirk Steinberg  
Lapishills Consulting Limited  
Cyprus

Email: dirk@lapishills.com

Robert Raszuk  
Bloomberg LP  
United States of America

Email: robert@raszuk.net

Bruno Decraene  
Orange  
France

Email: bruno.decraene@orange.com

Bart Peirens  
Proximus  
Belgium

Email: bart.peirens@proximus.com

Hani Elmalky  
Google  
United States of America

Email: helmalky@google.com

Prem Jonnalagadda  
Barefoot Networks  
United States of America

Email: prem@barefootnetworks.com

Milad Sharif  
SambaNova Systems  
United States of America

Email: milad.sharif@sambanova.ai

David Lebrun  
Google  
Belgium

Email: dlebrun@google.com

Stefano Salsano  
Universita di Roma "Tor Vergata"  
Italy

Email: stefano.salsano@uniroma2.it

Ahmed AbdelSalam  
Gran Sasso Science Institute  
Italy

Email: ahmed.abdelsalam@gssi.it

Gaurav Naik  
Drexel University  
United States of America

Email: gn@drexel.edu

Arthi Ayyangar  
Arrcus, Inc  
United States of America

Email: arthi@arrcus.com

Satish Mynam  
Arrcus, Inc  
United States of America

Email: satishm@arrcus.com

Wim Henderickx  
Nokia  
Belgium

Email: wim.henderickx@nokia.com

Shaowen Ma  
Juniper  
Singapore

Email: mashao@juniper.net

Ahmed Bashandy  
Individual  
United States of America

Email: abashandy.ietf@gmail.com

Francois Clad  
Cisco Systems, Inc.  
France

Email: fclad@cisco.com

Kamran Raza  
Cisco Systems, Inc.  
Canada

Email: skraza@cisco.com

Darren Dukes  
Cisco Systems, Inc.  
Canada

Email: ddukes@cisco.com

Patrice Brissette  
Cisco Systems, Inc.  
Canada

Email: pbrisset@cisco.com

Zafar Ali  
Cisco Systems, Inc.  
United States of America

Email: zali@cisco.com

Ketan Talaulikar  
Cisco Systems, Inc.  
India

Email: ketant@cisco.com

### 13. References

#### 13.1. Normative References

- [IEEE.802.3\_2018]  
IEEE, "802.3-2018", IEEE 802.3-2018,  
DOI 10.1109/IEEESTD.2018.8457469, August 2018,  
<<https://ieeexplore.ieee.org/document/8457469>>.
- [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>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in  
IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473,  
December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme,  
"IPv6 Flow Label Specification", RFC 6437,  
DOI 10.17487/RFC6437, November 2011,  
<<https://www.rfc-editor.org/info/rfc6437>>.
- [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>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6  
(IPv6) Specification", STD 86, RFC 8200,  
DOI 10.17487/RFC8200, July 2017,  
<<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,  
Decraene, B., Litkowski, S., and R. Shakir, "Segment  
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,  
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J.,  
Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header  
(SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020,  
<<https://www.rfc-editor.org/info/rfc8754>>.

## 13.2. Informative References

- [I-D.filsfils-spring-srv6-net-pgm-illustration]  
Filsfils, C., Camarillo, P., Li, Z., Matsushima, S., Decraene, B., Steinberg, D., Lebrun, D., Raszuk, R., and J. Leddy, "Illustrations for SRv6 Network Programming", draft-filsfils-spring-srv6-net-pgm-illustration-03 (work in progress), September 2020.
- [I-D.ietf-rtgwg-segment-routing-ti-lfa]  
Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-05 (work in progress), November 2020.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [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>>.
- [RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <<https://www.rfc-editor.org/info/rfc4664>>.
- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <<https://www.rfc-editor.org/info/rfc4761>>.
- [RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <<https://www.rfc-editor.org/info/rfc4762>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.
- [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>>.

- [RFC8214] Boutros, S., Sajassi, A., Salam, S., Drake, J., and J. Rabadan, "Virtual Private Wire Service Support in Ethernet VPN", RFC 8214, DOI 10.17487/RFC8214, August 2017, <<https://www.rfc-editor.org/info/rfc8214>>.
- [RFC8317] Sajassi, A., Ed., Salam, S., Drake, J., Uttaro, J., Boutros, S., and J. Rabadan, "Ethernet-Tree (E-Tree) Support in Ethernet VPN (EVPN) and Provider Backbone Bridging EVPN (PBB-EVPN)", RFC 8317, DOI 10.17487/RFC8317, January 2018, <<https://www.rfc-editor.org/info/rfc8317>>.

## Authors' Addresses

Clarence Filsfils (editor)  
Cisco Systems, Inc.  
Belgium

Email: [cf@cisco.com](mailto:cf@cisco.com)

Pablo Camarillo Garvia (editor)  
Cisco Systems, Inc.  
Spain

Email: [pcamaril@cisco.com](mailto:pcamaril@cisco.com)

John Leddy  
Individual Contributor  
United States of America

Email: [john@leddy.net](mailto:john@leddy.net)

Daniel Voyer  
Bell Canada  
Canada

Email: [daniel.voyer@bell.ca](mailto:daniel.voyer@bell.ca)

Satoru Matsushima  
SoftBank  
1-9-1, Higashi-Shimbashi, Minato-Ku  
Tokyo 105-7322  
Japan

Email: [satoru.matsushima@g.softbank.co.jp](mailto:satoru.matsushima@g.softbank.co.jp)

Zhenbin Li  
Huawei Technologies  
China

Email: [lizhenbin@huawei.com](mailto:lizhenbin@huawei.com)



INTAREA WG  
Internet-Draft  
Updates: 8335 (if approved)  
Intended status: Standards Track  
Expires: February 15, 2020

M. Nayak  
R. Bonica  
R. Puttur  
Juniper Networks  
August 16, 2019

Probing IP Interfaces By Vendor Specific Identifiers  
draft-nayak-intarea-probe-by-vfi-01

Abstract

This document enhances the PROBE diagnostic tool so that it can identify the probed interface by Vendor Specific Identifiers. In order to achieve that goal, this document also extends the Interface Identification Object. The Interface Identification Object is an ICMP Extension Object class.

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 February 15, 2019.

Copyright Notice

Copyright (c) 2019 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

1. Problem Statement . . . . .	2
2. Requirements Language . . . . .	2
3. ICMP Extended Echo Request Message . . . . .	3
4. Interface Identification Object . . . . .	3
5. ICMP Extended Echo Reply Message . . . . .	4
6. ICMP Message Processing . . . . .	4
7. Updates To RFC 8335 . . . . .	4
8. IANA Considerations . . . . .	5
9. Security Considerations . . . . .	5
10. Acknowledgements . . . . .	5
11. References . . . . .	5
11.1. Normative References . . . . .	5
11.2. Informative References . . . . .	5
Authors' Addresses . . . . .	6

## 1. Problem Statement

PROBE [RFC8335] is a diagnostic tool that can be used to query the status of an interface. PROBE sends an ICMP Extended Echo Request message to a proxy interface. The ICMP Extended Echo Request message contains an ICMP Extension Structure and the ICMP Extension Structure contains an Interface Identification Object. The Interface Identification Object identifies the probed interface by name, ifIndex or address.

When the proxy interface receives the ICMP Extended Echo Request, the node upon which it resides executes access control procedures as per [RFC8335] security considerations. If access is granted, the node determines the status of the probed interface and returns an ICMP Extended Echo Reply message. The ICMP Extended Echo Reply indicates the status of the probed interface.

Virtualized instance of the network adapter are created out of a Network Interface Card to enable efficient sharing of Network Interface Card in a virtualization environment. These Virtualized instances of the network adapter are implemented in the hardware and assigned a Vendor Specific Identifier to uniquely identify the Virtualized instance of the network adapter.

This document enhances the PROBE so that it can identify the probed interface by Vendor Specific Identifiers. This probe type is necessary when none of the other probe types of PROBE (i.e., probe interface by name, probe interface by address, etc) work. Virtual Function Index (VFI) [SR-IOV] is one (but not the only) instance of Vendor Specific Identifiers. Vendor Specific Identifiers, hereinafter referred to as VSI in the remaining part of this document. In order to achieve PROBE's enhancement, this document extends the Interface Identification Object. The Interface Identification Object is an ICMP Extension Object class.

## 2. Requirements Language

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. ICMP Extended Echo Request Message

Section 2 of [RFC8335] defines the ICMP Extended Echo Request message. As per [RFC8335], the ICMP Extended Echo Request message contains the following fields:

- o Type
- o Code
- o Checksum
- o Identifier
- o Reserved
- o L (local)
- o ICMP Extension Structure

Section 7 of [RFC4884] defines the ICMP Extension Structure. As per [RFC4884], the Extension Structure contains exactly one Extension Header followed by one or more objects. When applied to the ICMP Extended Echo Request message, the ICMP Extension Structure contains exactly one instance of the Interface Identification Object. Section 2.1 of [RFC8335] defines the Interface Identification Object. Section 4 of this document extends that definition.

If the L-bit is set, the Interface Identification Object can identify the probed interface by name, index, address or VFI. If the L-bit is clear, the Interface Identification Object identifies the probed interface by address.

### 4. Interface Identification Object

Section 2.1 of [RFC8335] defines the Interface Identification Object. The Interface Identification Object identifies the probed interface by name, index, or address. Like any other ICMP Extension Object, it contains an Object Header and Object Payload. The Object Header contains the following fields:

- o Class-Num: Interface Identification Object. The value is 3.
- o C-Type: Determines how the probed interface is identified.

- o Length: Length of the object, measured in octets, including the Object Header and Object Payload.

Currently, the following values are defined for C-Type:

- o (0) Reserved
- o (1) Identifies Interface by Name
- o (2) Identifies Interface by Index
- o (3) Identifies Interface by Address

This document defines the following, new C-Type:

- o (value TBD by IANA) Identifies Interfaces by Vendor Specific Identifier (VSI)

Every vendor specific ID needs to be N-bits long and the vendor gets to have exactly one of them (i.e, either VFI or some other Vendor Specific Identifiers). If the Interface Identification Object identifies the probed interface by VSI (Vendor Specific Identifier), the payload is as depicted in Figure 1.

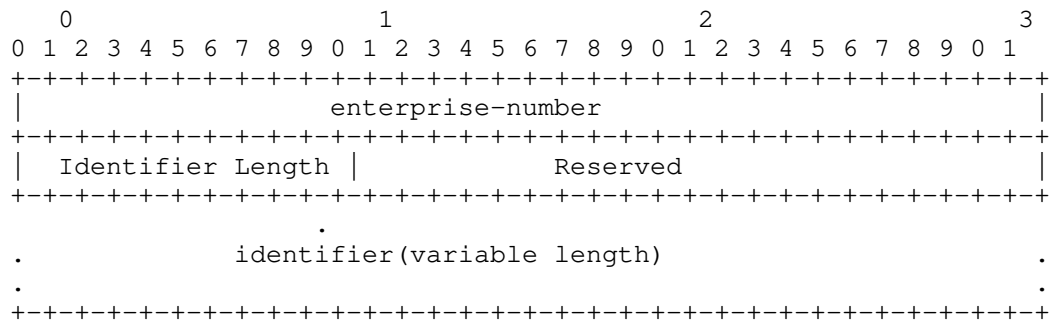


Figure 1: VSI Format

This VSI consists of the 4-octet vendor's registered Private Enterprise Number as maintained by IANA [IANA-PEN] followed by a unique identifier assigned by the vendor.

Payload fields are defined as follows:

- o Identifier Length: Number of significant bytes contained by the VSI. (The VSI field contains significant bytes and padding bytes.)
- o Reserved: This field MUST be set to 0 and ignored upon receipt.
- o Identifier: This variable-length field represents an Identifier associated with the probed interface. If the Identifier field would not otherwise terminate on a 32-bit boundary, it MUST be padded with zeroes.

## 5. ICMP Extended Echo Reply Message

Section 3 of [RFC8335] defines the ICMP Extended Echo Reply message. This document does not change that definition.

## 6. ICMP Message Processing

Section 4 of [RFC8335] defines the ICMP message processing. This document does not change that definition.

## 7. Updates To RFC 8335

Section 2 of [RFC8335] states:

"If the L-bit is set, the Interface Identification Object can identify the probed interface by name, index, or address. If the L-bit is clear, the Interface Identification Object MUST identify the probed interface by address."

This document updates that text as follows:

"If the L-bit is set, the Interface Identification Object can identify the probed interface by name, index, address, or Vendor Specific Identifier (VSI). If the L-bit is clear, the Interface Identification Object MUST identify the probed interface by address."

## 8. IANA Considerations

IANA is requested to add the following a new C-type:

- o (value TBD by IANA) Identifies Interfaces by Vendor Specific Identifier (VSI)

This new C-Type is to be added to the Interface Identification Object under the "ICMP Extension Object Classes and Class Sub-types" registry.

## 9. Security Considerations

This document neither extends nor mitigates any of the security considerations mentioned in [RFC8335].

## 10. Acknowledgements

The authors wish to acknowledge Ross Callon for his helpful comments.

## 11. References

### 11.1. Normative References

- [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>>.
- [RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <<https://www.rfc-editor.org/info/rfc4884>>.
- [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>>.

[RFC8335] Bonica, R., Thomas, R., Linkova, J., Lenart, C., and M. Boucadair, "PROBE: A Utility for Probing Interfaces", RFC 8335, DOI 10.17487/RFC8335, February 2018, <<https://www.rfc-editor.org/info/rfc8335>>

## 11.2. Informative References

[SR-IOV] PCI-SIG, ., "Single Root I/O Virtualization and Sharing Specification Revision 1.1", January 2010, <<https://members.pcisig.com/wg/PCI-SIG/document/download/8238>>.

## Authors' Addresses

Manoj Nayak  
Juniper Networks  
Bangalore, KA 560103  
India

Email: [manojnayak@juniper.net](mailto:manojnayak@juniper.net)

Ron Bonica  
Juniper Networks  
Herndon, Virginia 20171  
USA

Email: [rbonica@juniper.net](mailto:rbonica@juniper.net)

Rafik Puttur  
Juniper Networks  
Bangalore, KA 560103  
India

Email: [rafikp@juniper.net](mailto:rafikp@juniper.net)



Internet Area Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: May 7, 2020

V. Olteanu  
D. Niculescu  
University Politehnica of Bucharest  
November 04, 2019

SOCKS Protocol Version 6  
draft-olteanu-intarea-socks-6-08

Abstract

The SOCKS protocol is used primarily to proxy TCP connections to arbitrary destinations via the use of a proxy server. Under the latest version of the protocol (version 5), it takes 2 RTTs (or 3, if authentication is used) before data can flow between the client and the server.

This memo proposes SOCKS version 6, which reduces the number of RTTs used, takes full advantage of TCP Fast Open, and adds support for 0-RTT authentication.

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 May 7, 2020.

Copyright Notice

Copyright (c) 2019 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

1. Introduction . . . . .	3
1.1. Revision log . . . . .	4
2. Requirements language . . . . .	10
3. Mode of operation . . . . .	10
4. Requests . . . . .	11
5. Version Mismatch Replies . . . . .	13
6. Authentication Replies . . . . .	13
7. Operation Replies . . . . .	14
7.1. Handling CONNECT . . . . .	16
7.2. Handling BIND . . . . .	16
7.3. Handling UDP ASSOCIATE . . . . .	16
7.3.1. Proxying UDP servers . . . . .	19
7.3.2. Proxying multicast traffic . . . . .	19
8. SOCKS Options . . . . .	19
8.1. Stack options . . . . .	20
8.1.1. IP TOS options . . . . .	21
8.1.2. Happy Eyeballs options . . . . .	21
8.1.3. TFO options . . . . .	22
8.1.4. Multipath options . . . . .	22
8.1.5. Listen Backlog options . . . . .	23
8.2. Authentication Method options . . . . .	24
8.3. Authentication Data options . . . . .	25
8.4. Session options . . . . .	26
8.4.1. Session initiation . . . . .	26
8.4.2. Further SOCKS Requests . . . . .	27
8.4.3. Tearing down the session . . . . .	28
8.5. Idempotence options . . . . .	29
8.5.1. Requesting a token window . . . . .	29
8.5.2. Spending a token . . . . .	30
8.5.3. Shifting windows . . . . .	32
8.5.4. Out-of-order Window Advertisements . . . . .	32
9. Username/Password Authentication . . . . .	32
10. TCP Fast Open on the Client-Proxy Leg . . . . .	33
11. False Starts . . . . .	33
12. DNS provided by SOCKS . . . . .	34
13. Security Considerations . . . . .	34
13.1. Large requests . . . . .	34
13.2. Replay attacks . . . . .	35
13.3. Resource exhaustion . . . . .	35
14. IANA Considerations . . . . .	35
15. Acknowledgements . . . . .	36

16. References . . . . .	36
16.1. Normative References . . . . .	36
16.2. Informative References . . . . .	37
Authors' Addresses . . . . .	37

## 1. Introduction

Versions 4 and 5 [RFC1928] of the SOCKS protocol were developed two decades ago and are in widespread use for circuit level gateways or as circumvention tools, and enjoy wide support and usage from various software, such as web browsers, SSH clients, and proxifiers. However, their design needs an update in order to take advantage of the new features of transport protocols, such as TCP Fast Open [RFC7413], or to better assist newer transport protocols, such as MPTCP [RFC6824].

One of the main issues faced by SOCKS version 5 is that, when taking into account the TCP handshake, method negotiation, authentication, connection request and grant, it may take up to 5 RTTs for a data exchange to take place at the application layer. This is especially costly in networks with a large delay at the access layer, such as 3G, 4G, or satellite.

The desire to reduce the number of RTTs manifests itself in the design of newer security protocols. TLS version 1.3 [RFC8446] defines a zero round trip (0-RTT) handshake mode for connections if the client and server had previously communicated.

TCP Fast Open [RFC7413] is a TCP option that allows TCP to send data in the SYN and receive a response in the first ACK, and aims at obtaining a data response in one RTT. The SOCKS protocol needs to concern itself with at least two TFO deployment scenarios: First, when TFO is available end-to-end (at the client, at the proxy, and at the server); second, when TFO is active between the client and the proxy, but not at the server.

This document describes the SOCKS protocol version 6. The key improvements over SOCKS version 5 are:

- o The client sends as much information upfront as possible, and does not wait for the authentication process to conclude before requesting the creation of a socket.
- o The connection request also mimics the semantics of TCP Fast Open [RFC7413]. As part of the connection request, the client can supply the potential payload for the initial SYN that is sent out to the server.

- o The protocol can be extended via options without breaking backward-compatibility.
- o The protocol can leverage the aforementioned options to support 0-RTT authentication schemes.

### 1.1. Revision log

Typos and minor clarifications are not listed.

draft-08

- o Removed Address Resolution options
- o Happy Eyeballs options
- o DNS provided by SOCKS

draft-07

- o All fields are now aligned.
- o Eliminated version minors
- o Lots of changes to options
  - \* 2-byte option kinds
  - \* Flattened option kinds/types/reply codes; also renamed some options
  - \* Socket options
    - + Proxies MUST always answer them (Clients can probe for support)
    - + MPTCP Options: expanded functionality ("please do/don't do MPTCP on my behalf")
    - + MPTCP Scheduler options removed
    - + Listen Backlog options: code changed to 0x03
  - \* Revamped Idempotence options
  - \* Auth data options limited to one per method

- o Authentication Reply: all authentication-related information is now in the options
  - \* Authentication replies no longer have a field indicating the chosen auth. method
  - \* Method that must proceed (or whereby authentication succeeded) indicated in options
  - \* Username/password authentication: proxy now sends reply in option
- o Removed requirements w.r.t. caching authentication methods by multihomed clients
- o UDP: 8-byte association IDs
- o Sessions
  - \* The proxy is now free to terminate ongoing connections along with the session.
  - \* The session-terminating request is not part of the session that it terminated.
- o Address Resolution options

draft-06

- o Session options
- o Options now have a 2-byte length field.
- o Stack options
  - \* Stack options can no longer contain duplicate information.
  - \* TFO: Better payload size semantics
  - \* TOS: Added missing code field.
  - \* MPTCP Scheduler options:
    - + Removed support for round-robin
    - + "Default" renamed to "Lowest latency first"

- \* Listen Backlog options: now tied to sessions, instead of an authenticated user
- o Idempotence options
  - \* Now used in the context of a session (no longer tied to an authenticated user)
  - \* Idempotence options have a different codepoint: 0x05. (Was 0x04.)
  - \* Clarified that implementations that support Idempotence Options must support all Idempotence Option Types.
  - \* Shifted Idempotence Option Types by 1. (Makes implementation easier.)
- o Shrunk vendor-specific option range to 32 (down from 64).
- o Removed reference to dropping initial data. (It could no longer be done as of -05.)
- o Initial data size capped at 16KB.
- o Application data is never encrypted by SOCKS 6. (It can still be encrypted by the TLS layer under SOCKS.)
- o Messages now carry the total length of the options, rather than the number of options. Limited options length to 16KB.
- o Security Considerations
  - \* Updated the section to reflect the smaller maximum message size.
  - \* Added a subsection on resource exhaustion.

draft-05

- o Limited the "slow" authentication negotiations to one (and Authentication Replies to 2)
- o Revamped the handling of the first bytes in the application data stream
  - \* False starts are now recommended. (Added the "False Start" section.)

- \* Initial data is only available to clients willing to do "slow" authentication. Moved the "Initial data size" field from Requests to Authentication Method options.
  - \* Initial data size capped at  $2^{13}$ . Initial data can no longer be dropped by the proxy.
  - \* The TFO option can hint at the desired SYN payload size.
  - o Request: clarified the meaning of the Address and Port fields.
  - o Better reverse TCP proxy support: optional listen backlog for TCP BIND
  - o TFO options can no longer be placed inside Operation Replies.
  - o IP TOS stack option
  - o Suggested a range for vendor-specific options.
  - o Revamped UDP functionality
    - \* Now using fixed UDP ports
    - \* DTLS support
  - o Stack options: renamed Proxy-Server leg to Proxy-Remote leg
- draft-04
- o Moved Token Expenditure Replies to the Authentication Reply.
  - o Shifted the Initial Data Size field in the Request, in order to make it easier to parse.

draft-03

- o Shifted some fields in the Operation Reply to make it easier to parse.
- o Added connection attempt timeout response code to Operation Replies.
- o Proxies send an additional Authentication Reply after the authentication phase. (Useful for token window advertisements.)
- o Renamed the section "Connection Requests" to "Requests"

- o Clarified the fact that proxies don't need to support any command in particular.
- o Added the section "TCP Fast Open on the Client-Proxy Leg"
- o Options:
  - \* Added constants for option kinds
  - \* Salt options removed, along with the relevant section from Security Considerations. (TLS 1.3 Makes AEAD mandatory.)
  - \* Limited Authentication Data options to one per method.
  - \* Relaxed proxy requirements with regard to handling multiple Authentication Data options. (When the client violates the above bullet point.)
  - \* Removed interdependence between Authentication Method and Authentication Data options.
  - \* Clients SHOULD omit advertising the "No authentication required" option. (Was MAY.)
  - \* Idempotence options:
    - + Token Window Advertisements are now part of successful Authentication Replies (so that the proxy-server RTT has no impact on their timeliness).
    - + Proxies can't advertise token windows of size 0.
    - + Tweaked token expenditure response codes.
    - + Support no longer mandatory on the proxy side.
  - \* Revamped Socket options
    - + Renamed Socket options to Stack options.
    - + Banned contradictory socket options.
    - + Added socket level for generic IP. Removed the "socket" socket level.
    - + Stack options no longer use option codes from `setsockopt()`.
    - + Changed MPTCP Scheduler constants.



draft-02

- o Made support for Idempotence options mandatory for proxies.
- o Clarified what happens when proxies can not or will not issue tokens.
- o Limited token windows to  $2^{31} - 1$ .
- o Fixed definition of "less than" for tokens.
- o NOOP commands now trigger Operation Replies.
- o Renamed Authentication options to Authentication Data options.
- o Authentication Data options are no longer mandatory.
- o Authentication methods are now advertised via options.
- o Shifted some Request fields.
- o Option range for vendor-specific options.
- o Socket options.
- o Password authentication.
- o Salt options.

draft-01

- o Added this section.
- o Support for idempotent commands.
- o Removed version numbers from operation replies.
- o Request port number for SOCKS over TLS. Deprecate encryption/encapsulation within SOCKS.
- o Added Version Mismatch Replies.
- o Renamed the AUTH command to NOOP.
- o Shifted some fields to make requests and operation replies easier to parse.

## 2. 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 [RFC2119].

## 3. Mode of operation

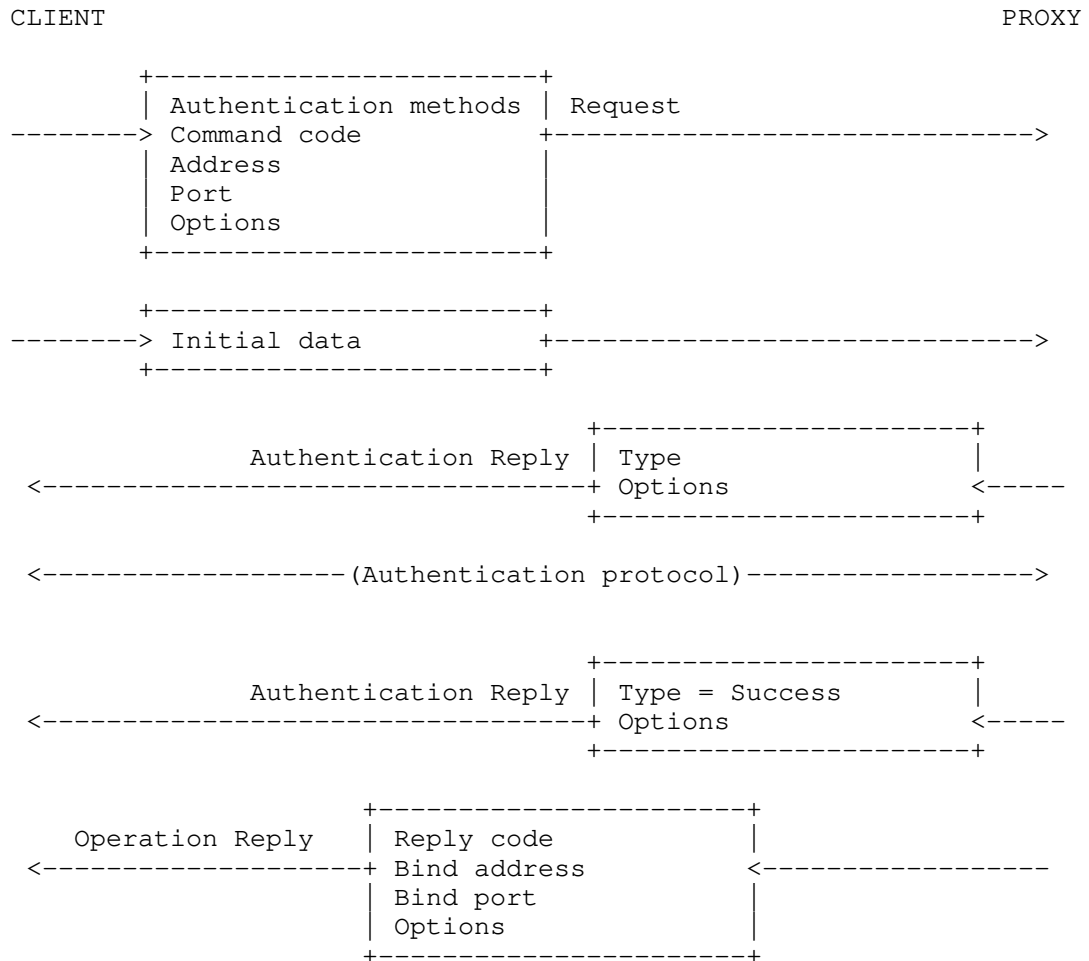


Figure 1: The SOCKS version 6 protocol message exchange

When a TCP-based client wishes to establish a connection to a server, it must open a TCP connection to the appropriate SOCKS port on the

SOCKS proxy. The client then enters a negotiation phase, by sending the request in Figure 1, that contains, in addition to fields present in SOCKS 5 [RFC1928], fields that facilitate low RTT usage and faster authentication negotiation.

Next, the server sends an authentication reply. If the request did not contain the necessary authentication information, the proxy indicates an authentication method that must proceed. This may trigger a longer authentication sequence that could include tokens for ulterior faster authentications. The part labeled "Authentication protocol" is specific to the authentication method employed and is not expected to be employed for every connection between a client and its proxy server. The authentication protocol typically takes up 1 RTT or more.

If the authentication is successful, an operation reply is generated by the proxy. It indicates whether the proxy was successful in creating the requested socket or not.

In the fast case, when authentication is properly set up, the proxy attempts to create the socket immediately after the receipt of the request, thus achieving an operational connection in one RTT (provided TFO functionality is available at the client, proxy, and server).

#### 4. Requests

The client starts by sending a request to the proxy.

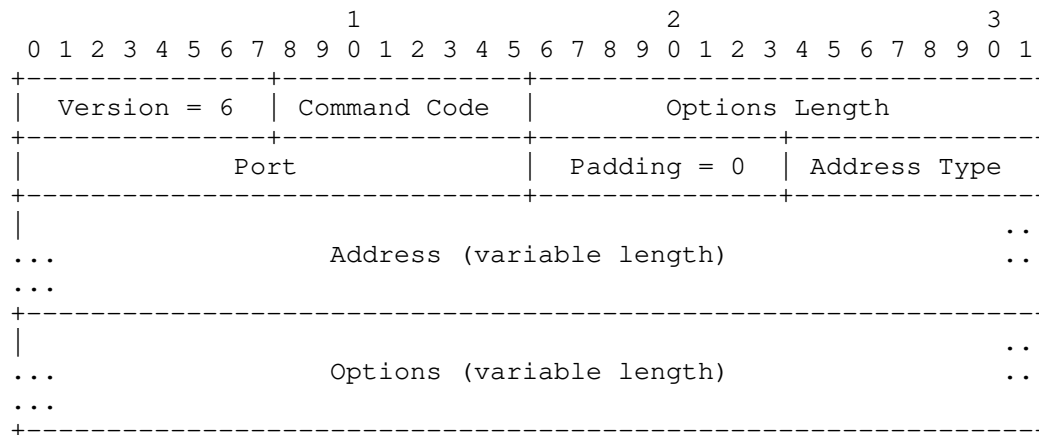


Figure 2: SOCKS 6 Request

- o Version: 6

- o Command Code:
  - \* 0x00 NOOP: does nothing.
  - \* 0x01 CONNECT: requests the establishment of a TCP connection. TFO MUST NOT be used unless explicitly requested.
  - \* 0x02 BIND: requests the establishment of a TCP port binding.
  - \* 0x03 UDP ASSOCIATE: requests a UDP port association.
- o Address Type:
  - \* 0x01: IPv4
  - \* 0x03: Domain Name
  - \* 0x04: IPv6
- o Address: this field's format depends on the address type:
  - \* IPv4: a 4-byte IPv4 address
  - \* Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated, but padded by NUL characters, if needed.
  - \* IPv6: a 16-byte IPv6 address
- o Port: the port in network byte order.
- o Padding: set to 0
- o Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.
- o Options: see Section 8.

The Address and Port fields have different meanings based on the Command Code:

- o NOOP: The fields have no meaning. The Address Type field MUST be either 0x01 (IPv4) or 0x04 (IPv6). The Address and Port fields MUST be 0.
- o CONNECT: The fields signify the address and port to which the client wishes to connect.

- o BIND, UDP ASSOCIATE: The fields indicate the desired bind address and port. If the client does not require a certain address, it can set the Address Type field to 0x01 (IPv4) or 0x04 (IPv6), and the Address field to 0. Likewise, if the client does not require a certain port, it can set the Port field to 0.

Clients can advertise their supported authentication methods by including an Authentication Method Advertisement option (see Section 8.2).

## 5. Version Mismatch Replies

Upon receipt of a request starting with a version number other than 6, the proxy sends the following response:

```

 0 1 2 3 4 5 6 7
+-----+
| Version = 6 |
+-----+

```

Figure 3: SOCKS 6 Version Mismatch Reply

- o Version: 6

A client **MUST** close the connection after receiving such a reply.

## 6. Authentication Replies

Upon receipt of a valid request, the proxy sends an Authentication Reply:

										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Version = 6										Type										Options Length																			
										Options (variable length)																													

Figure 4: SOCKS 6 Authentication Reply

- o Version: 6

- o Type:
  - \* 0x00: authentication successful.
  - \* 0x01: authentication failed.
- o Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.
- o Options: see Section 8.

If the server signals that the authentication has failed and does not signal that any authentication negotiation can continue (via an Authentication Method Selection option), the client MUST close the connection.

The client and proxy begin a method-specific negotiation. During such negotiations, the proxy MAY supply information that allows the client to authenticate a future request using an Authentication Data option. Application data is not subject to any encryption negotiated during this phase. Descriptions of such negotiations are beyond the scope of this memo.

When the negotiation is complete (either successfully or unsuccessfully), the proxy sends a second Authentication Reply. The second Authentication Reply MUST NOT allow for further negotiations.

## 7. Operation Replies

After the authentication negotiations are complete, the proxy sends an Operation Reply:

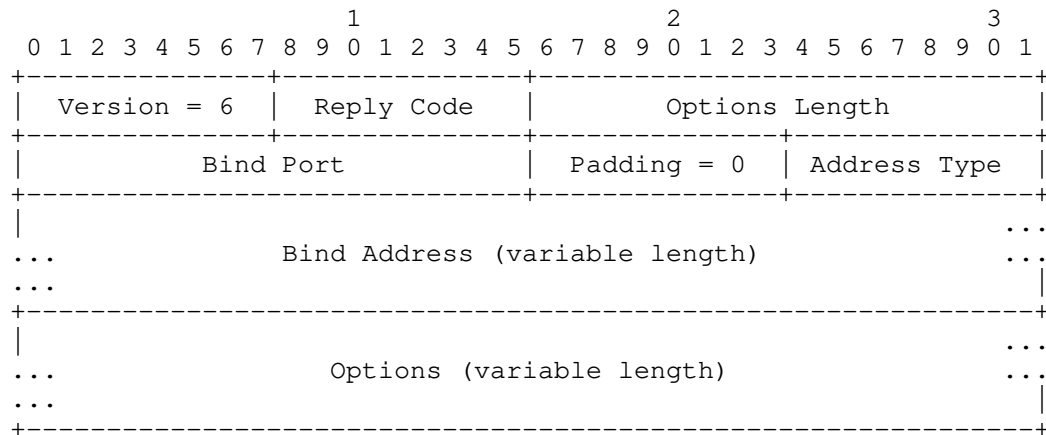


Figure 5: SOCKS 6 Operation Reply

- o Version: 6
- o Reply Code:
  - \* 0x00: Success
  - \* 0x01: General SOCKS server failure
  - \* 0x02: Connection not allowed by ruleset
  - \* 0x03: Network unreachable
  - \* 0x04: Host unreachable
  - \* 0x05: Connection refused
  - \* 0x06: TTL expired
  - \* 0x07: Command not supported
  - \* 0x08: Address type not supported
  - \* 0x09: Connection attempt timed out
- o Bind Port: the proxy bound port in network byte order.
- o Padding: set to 0
- o Address Type:

- \* 0x01: IPv4
- \* 0x03: Domain Name
- \* 0x04: IPv6
- o Bind Address: the proxy bound address in the following format:
  - \* IPv4: a 4-byte IPv4 address
  - \* Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated, but padded by NUL characters, if needed.
  - \* IPv6: a 16-byte IPv6 address
- o Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.
- o Options: see Section 8.

Proxy implementations MAY support any subset of the client commands listed in Section 4.

If the proxy returns a reply code other than "Success", the client MUST close the connection.

If the client issued an NOOP command, the client MUST close the connection after receiving the Operation Reply.

### 7.1. Handling CONNECT

In case the client has issued a CONNECT request, data can now pass.

### 7.2. Handling BIND

In case the client has issued a BIND request, it must wait for a second Operation reply from the proxy, which signifies that a host has connected to the bound port. The Bind Address and Bind Port fields contain the address and port of the connecting host. Afterwards, application data may pass.

### 7.3. Handling UDP ASSOCIATE

Proxies offering UDP functionality must be configured with a UDP port used for relaying UDP datagrams to and from the client, and/or a port used for relaying datagrams over DTLS.



Following a successful Operation Reply, the proxy sends a UDP Association Initialization message:

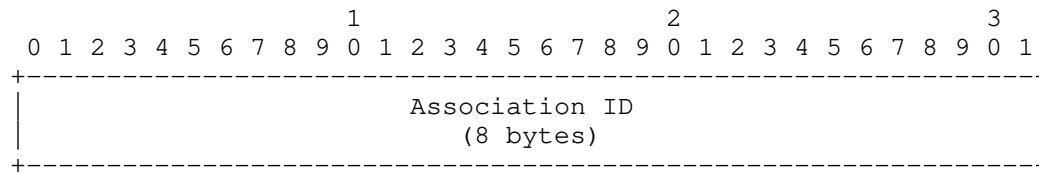


Figure 6: UDP Association Initialization

- o Association ID: the identifier of the UDP association

Proxy implementations SHOULD generate Association IDs randomly or pseudo-randomly.

Clients may start sending UDP datagrams to the proxy either in plaintext, or over an established DTLS session, using the proxy's configured UDP ports. A client's datagrams are prefixed by a SOCKS Datagram Header, indicating the remote host's address and port:

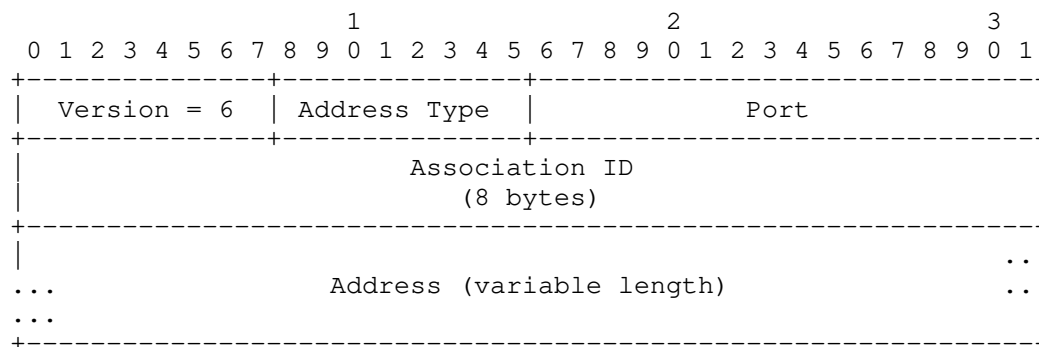


Figure 7: SOCKS 6 Datagram Header

- o Version: 0x06
- o Association ID: the identifier of the UDP association
- o Address Type:
  - \* 0x01: IPv4
  - \* 0x03: Domain Name

- \* 0x04: IPv6
- o Address: this field's format depends on the address type:
  - \* IPv4: a 4-byte IPv4 address
  - \* Domain Name: one byte that contains the length of the FQDN, followed by the FQDN itself. The string is not NUL-terminated.
  - \* IPv6: a 16-byte IPv6 address
- o Port: the port in network byte order.

Following the receipt of the first datagram from the client, the proxy makes a one-way mapping between the Association ID and:

- o the 5-tuple of the UDP conversation, if the datagram was received over plain UDP, or
- o the DTLS connection, if the datagram was received over DTLS. The DTLS connection is identified either by its 5-tuple, or some other mechanism, like [I-D.ietf-tls-dtls-connection-id].

Further datagrams carrying the same Association ID, but not matching the established mapping, are silently dropped.

The proxy then sends an UDP Association Confirmation message over the TCP connection with the client:

```

 0 1 2 3 4 5 6 7
+-----+
| Status = 0 |
+-----+
```

Figure 8: UDP Association Confirmation

- o Status: MUST be 0x00

Following the confirmation message, UDP packets bound for the proxy's bind address and port are relayed to the client, also prefixed by a Datagram Header.

The UDP association remains active for as long as the TCP connection between the client and the proxy is kept open.

### 7.3.1. Proxying UDP servers

Under some circumstances (e.g. when hosting a server), the SOCKS client expects the remote host to send UDP datagrams first. As such, the SOCKS client must trigger a UDP Association Confirmation without having the proxy relay any datagrams on its behalf.

To that end, it sends an empty datagram prefixed by a Datagram Header with an IP address and port consisting of zeroes. The client **SHOULD** resend the empty datagram if an UDP Association Confirmation is not received after a timeout.

### 7.3.2. Proxying multicast traffic

The use of multicast addresses is permitted for UDP traffic only.

## 8. SOCKS Options

SOCKS options have the following format:

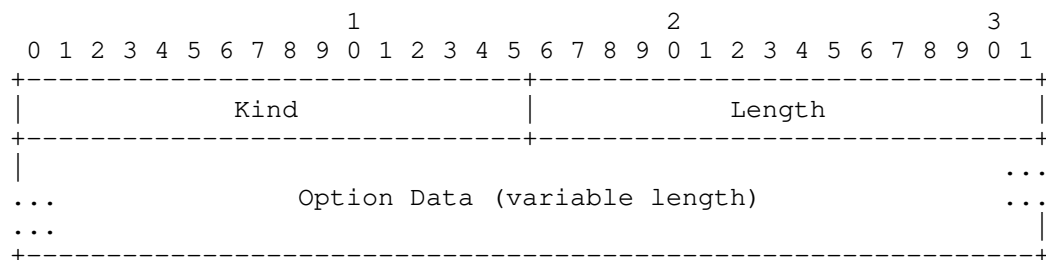


Figure 9: SOCKS 6 Option

- o Kind: Allocated by IANA. (See Section 14.)
- o Length: The total length of the option. **MUST** be a multiple of 4.
- o Option Data: The contents are specific to each option kind.

Unless otherwise noted, client and proxy implementations **MAY** omit supporting any of the options described in this document. Upon encountering an unsupported option, a SOCKS endpoint **MUST** silently ignore it.

### 8.1. Stack options

Stack options can be used by clients to alter the behavior of the protocols on top of which SOCKS is running, as well the protocols used by the proxy to communicate with the remote host (i.e. IP, TCP, UDP). A Stack option can affect either the proxy's protocol on the client-proxy leg or on the proxy-remote leg. Clients can only place Stack options inside SOCKS Requests.

Proxies MAY choose not to honor any Stack options sent by the client.

Proxies include Stack options in their Operation Replies to signal their behavior, and MUST do so for every supported Stack option sent by the client. Said options MAY also be unsolicited, i. e. the proxy MAY send them to signal behaviour that was not explicitly requested by the client.

If a particular Stack option is unsupported, the proxy MUST silently ignore it.

In case of UDP ASSOCIATE, the stack options refer to the UDP traffic relayed by the proxy.

Stack options that are part of the same message MUST NOT contradict one another or contain duplicate information.

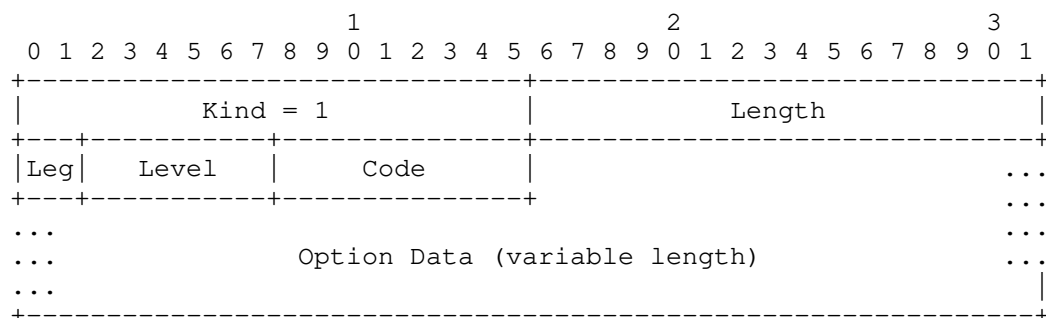


Figure 10: Stack Option

o Leg:

- \* 1: Client-Proxy Leg
- \* 2: Proxy-Remote Leg
- \* 3: Both Legs

- o Level:
  - \* 1: IP: options that apply to either IPv4 or IPv6
  - \* 2: IPv4
  - \* 3: IPv6
  - \* 4: TCP
  - \* 5: UDP
- o Code: Option code
- o Option Data: Option-specific data

#### 8.1.1. IP TOS options

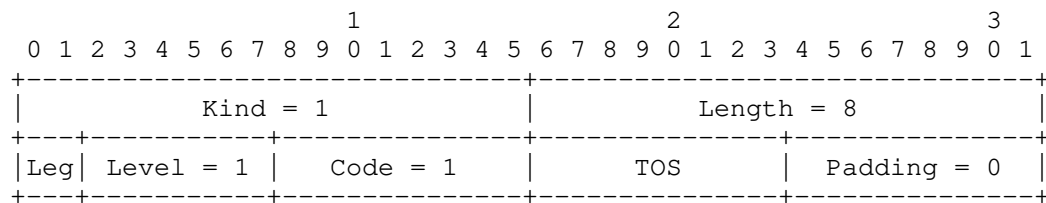


Figure 11: IP TOS Option

- o TOS: The IP TOS code

The client can use IP TOS options to request that the proxy use a certain value for the IP TOS field. Likewise, the proxy can use IP TOS options to advertise the TOS values being used.

#### 8.1.2. Happy Eyeballs options

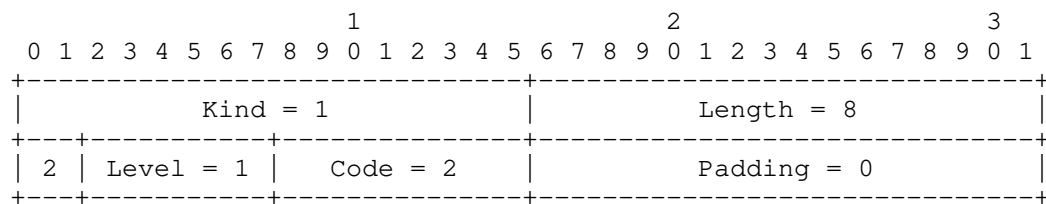


Figure 12: Happy Eyeballs Option

This memo provides enough features for clients to implement a mechanism analogous to Happy Eyeballs [RFC8305] over SOCKS. However, when the delay between the client and the proxy, or the proxy's vantage point, is high, doing so can become impractical or inefficient.

In such cases, the client can instruct the proxy to employ the Happy Eyeballs technique on its behalf when connecting to a remote host.

The client **MUST** supply a Domain Name as part of its Request. Otherwise, the proxy **MUST** silently ignore the option.

TODO: Figure out which knobs to include.

#### 8.1.3. TFO options

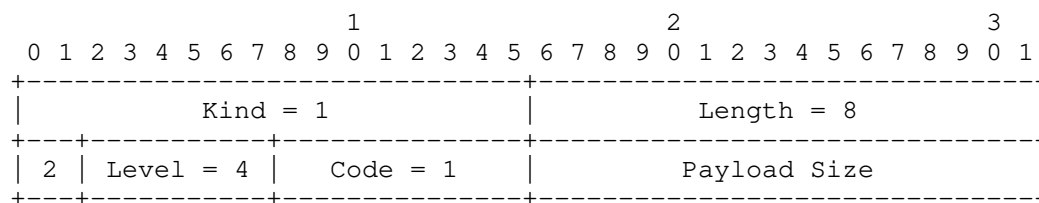


Figure 13: TFO Option

- o Payload Size: The desired payload size of the TFO SYN. Ignored in case of a BIND command.

If a SOCKS Request contains a TFO option, the proxy **SHOULD** attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command. Otherwise, the proxy **MUST NOT** attempt to use TFO in case of a CONNECT command, or accept TFO in case of a BIND command.

In case of a CONNECT command, the client can indicate the desired payload size of the SYN. If the field is 0, the proxy can use an arbitrary payload size. If the field is non-zero, the proxy **MUST NOT** use a payload size larger than the one indicated. The proxy **MAY** use a smaller payload size than the one indicated.

#### 8.1.4. Multipath options

In case of a CONNECT or BIND command, the client can inform the proxy whether MPTCP is desired on the proxy-remote leg by sending a Multipath option.

Conversely, the proxy can use a Multipath option to convey the following information: \* whether or not the connection uses MPTCP or not, when replying to a CONNECT command, or in the second Operation reply to a BIND command, or \* whether an MPTCP connection will be accepted, when first replying to a BIND command.

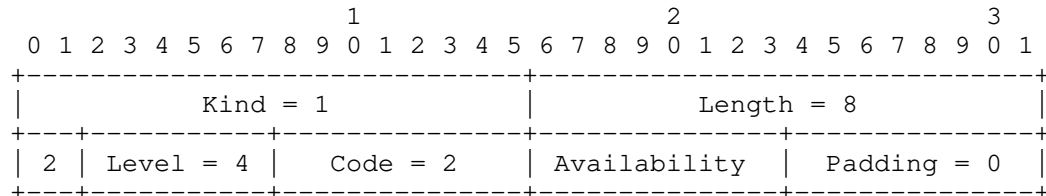


Figure 14: Multipath Option

o Availability:

- \* 0x01: MPTCP is not desired or available
- \* 0x02: MPTCP is desired or available

In the absence of such an option, the proxy SHOULD NOT enable MPTCP.

#### 8.1.1.5. Listen Backlog options

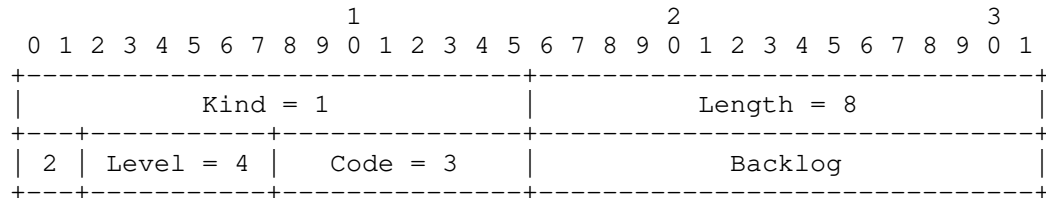


Figure 15: Listen Backlog Option

o Backlog: The length of the listen backlog.

The default behavior of the BIND does not allow a client to simultaneously handle multiple connections to the same bind address. A client can alter BIND's behavior by adding a TCP Listen Backlog Option to a BIND Request, provided that the Request is part of a Session.

In response, the proxy sends a TCP Listen Backlog Option as part of the Operation Reply, with the Backlog field signalling the actual

backlog used. The proxy SHOULD NOT use a backlog longer than requested.

Following the successful negotiation of a backlog, the proxy listens for incoming connections for as long as the initial connection stays open. The initial connection is not used to relay data between the client and a remote host.

To accept connections, the client issues further BIND Requests using the bind address and port supplied by the proxy in the initial Operation Reply. Said BIND requests must belong to the same Session as the original Request.

If no backlog is issued, the proxy signals a backlog length of 0, and BIND's behavior remains unaffected.

## 8.2. Authentication Method options

A client that is willing to go through the authentication phase MUST include an Authentication Method Advertisement option in its Request. In case of a CONNECT Request, the option is also used to specify the amount of initial data supplied before any method-specific authentication negotiations take place.

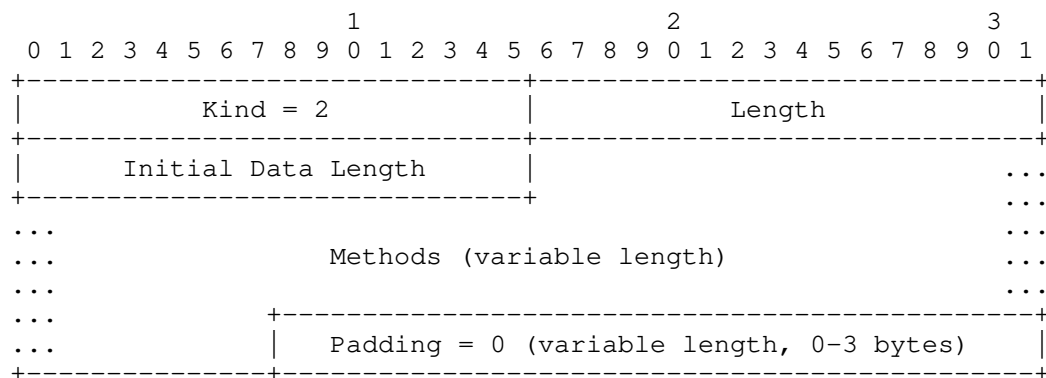


Figure 16: Authentication Method Advertisement Option

- o Initial Data Size: A two-byte number in network byte order. In case of CONNECT, this is the number of bytes of initial data that are supplied by the client immediately following the Request. This number MUST NOT be larger than  $2^{14}$ .
- o Methods: One byte per advertised method. Method numbers are assigned by IANA.



- o **Padding:** A minimally-sized sequence of zeroes, such that the option length is a multiple of 4. Note that 0 coincides with the value for "No Authentication Required".

Clients **MUST** support the "No authentication required" method. Clients **SHOULD** omit advertising the "No authentication required" option.

The proxy indicates which authentication method must proceed by sending an Authentication Method Selection option in the corresponding Authentication Reply:

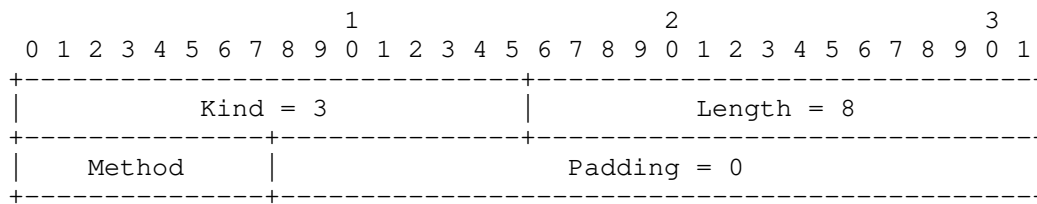


Figure 17: Authentication Method Selection Option

- o **Method:** The selected method.

If the proxy selects "No Acceptable Methods", the client **MUST** close the connection.

If authentication is successful via some other means, or not required at all, the proxy silently ignores the Authentication Method Advertisement option.

### 8.3. Authentication Data options

Authentication Data options carry method-specific authentication data. They can be part of SOCKS Requests and Authentication Replies.

Authentication Data options have the following format:

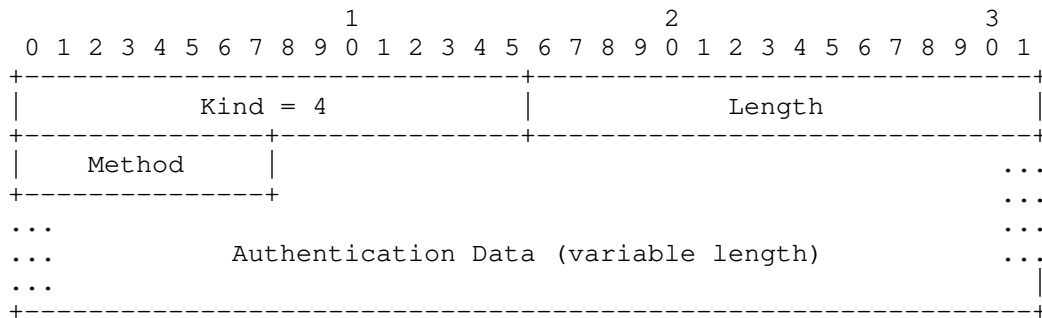


Figure 18: Authentication Data Option

- o Method: The number of the authentication method. These numbers are assigned by IANA.
- o Authentication Data: The contents are specific to each method.

Clients MUST only place one Authentication Data option per authentication method.

#### 8.4. Session options

Clients and proxies can establish SOCKS sessions, which span one or more Requests. All session-related negotiations are done via Session Options, which are placed in Requests and Authentication Replies by the client and, respectively, by the proxy.

Client and proxy implementations MUST either support all Session Option Types, or none.

##### 8.4.1. Session initiation

A client can initiate a session by sending a Session Request Option:

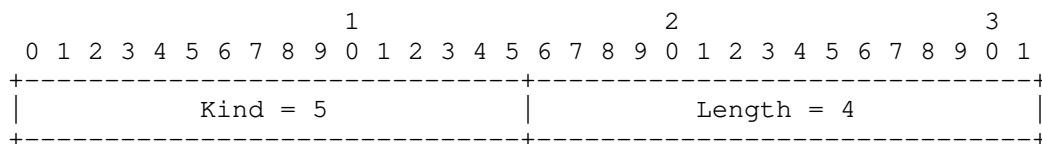


Figure 19: Session Request Option

The proxy then replies with a Session ID Option in the successful Operation Reply:

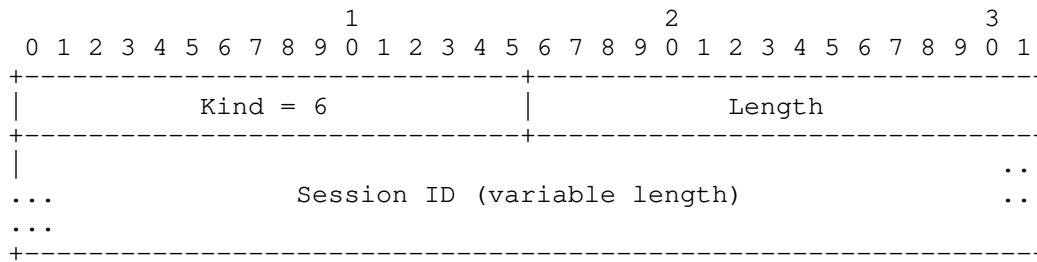


Figure 20: Session ID Option

- o Session ID: An opaque sequence of bytes specific to the session. The size MUST be a multiple of 4. MUST NOT be empty.

The Session ID serves to identify the session and is opaque to the client.

The credentials, or lack thereof, used to initiate the session are tied to the session. If authentication is to be performed for further Requests, the session is deemed "untrusted", and the proxy also places a Session Untrusted option in the Authentication Reply:

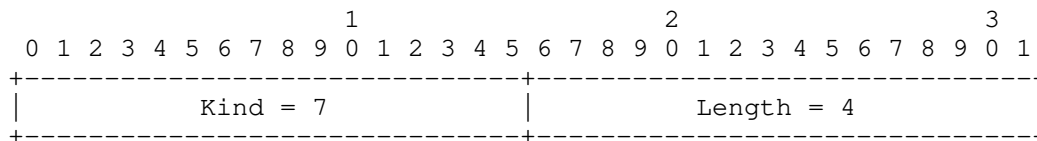


Figure 21: Session Untrusted Option

The SOCKS Request that initiated the session is considered part of the session. A client MUST NOT attempt to initiate a session from within a different session.

If the proxy can not or will not honor the Session Request, it does so silently.

#### 8.4.2. Further SOCKS Requests

Any further SOCKS Requests that are part of the session MUST include a Session ID Option (as seen in Figure 20).

The authentication procedure is altered based on the Session ID's validity and whether or not the Session is untrusted.

For valid Session IDs:

- o If the session is untrusted, the proxy MUST reject clients that do not authenticate using the same method and credentials that were used to initiate the session.
- o Otherwise, the proxy MUST silently ignore any authentication attempt in the Request, and MUST NOT require any authentication.

The proxy then replies by placing a Session OK option in the successful Authentication Reply:

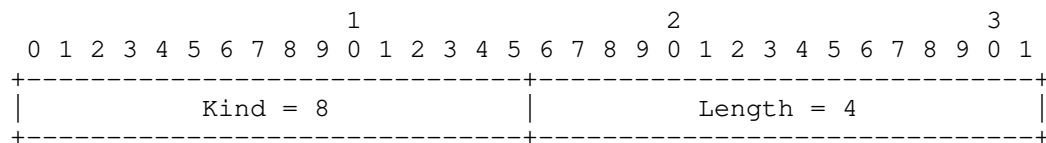


Figure 22: Session OK Option

If the ticket is invalid, the first Authentication Reply MUST signal that authentication failed and can not continue (by setting the Type field to 0x01). Further, it SHALL contain a Session Invalid option:

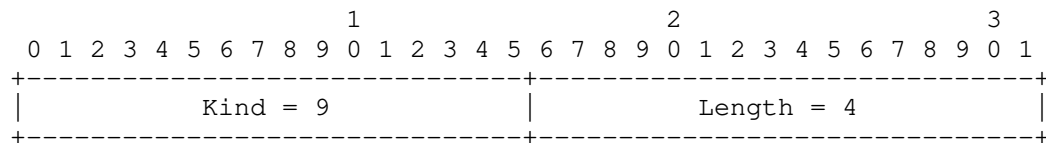


Figure 23: Session Invalid Option

#### 8.4.3. Tearing down the session

Proxies can, at their discretion, tear down a session and free all associated state. Proxy implementations SHOULD feature a timeout mechanism that destroys sessions after a period of inactivity. When a session is terminated, the proxy MAY close all connections associated with said session.

Clients can signal that a session is no longer needed, and can be torn down, by sending a Session Teardown option in addition to the Session ID option:

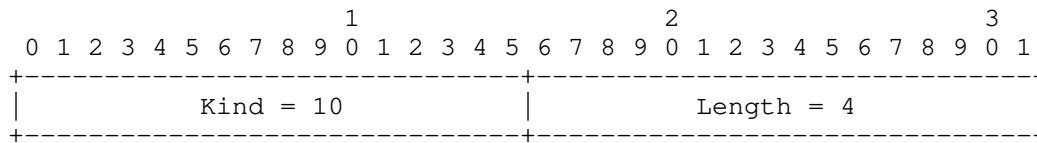


Figure 24: Session Teardown Option

After sending such an option, the client **MUST** assume that the session is no longer valid. The proxy **MUST** treat the connection-terminating request as if it were not part of any session.

#### 8.5. Idempotence options

To protect against duplicate SOCKS Requests, clients can request, and then spend, idempotence tokens. A token can only be spent on a single SOCKS request.

Tokens are 4-byte unsigned integers in a modular 4-byte space. Therefore, if  $x$  and  $y$  are tokens,  $x$  is less than  $y$  if  $0 < (y - x) < 2^{31}$  in unsigned 32-bit arithmetic.

Proxies grant contiguous ranges of tokens called token windows. Token windows are defined by their base (the first token in the range) and size.

All token-related operations are done via Idempotence options.

Idempotence options are only valid in the context of a SOCKS Session. If a SOCKS Request is not part of a Session (either by supplying a valid Session ID or successfully initiating one via a Session Request), the proxy **MUST** silently ignore any Idempotence options.

Token windows are tracked by the proxy on a per-session basis. There can be at most one token window for every session and its tokens can only be spent from within said session.

Client and proxy implementations **MUST** either support all Idempotence Option Types, or none.

##### 8.5.1. Requesting a token window

A client can obtain a window of tokens by sending an Idempotence Request option as part of a SOCKS Request:

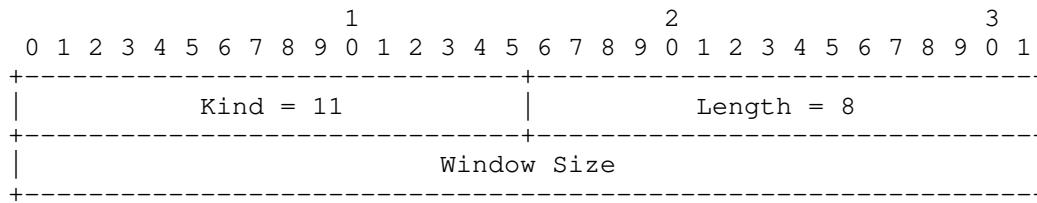


Figure 25: Token Request

- o Window Size: The requested window size.

Once a token window is issued, the proxy **MUST** include an Idempotence Window option in all subsequent successful Authentication Replies:

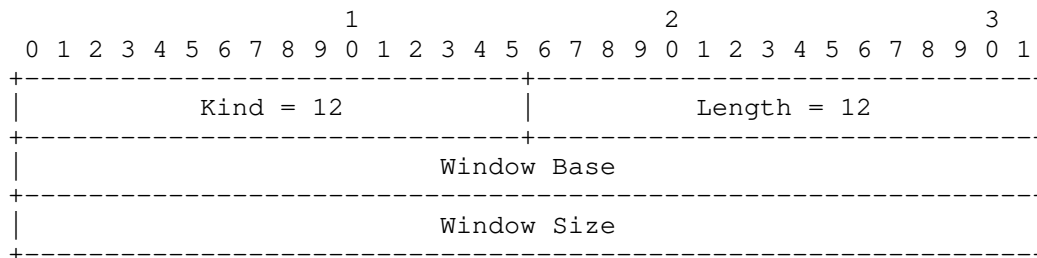


Figure 26: Idempotence Window

- o Window Base: The first token in the window.
- o Window Size: The window size. This value **MAY** differ from the requested window size. Window sizes **MUST** be less than  $2^{31}$ . Window sizes **MUST NOT** be 0.

If no token window is issued, the proxy **MUST** silently ignore the Token Request. If there is already a token window associated with the session, the proxy **MUST NOT** issue a new window.

#### 8.5.2. Spending a token

The client can attempt to spend a token by including a Idempotence Expenditure option in its SOCKS request:

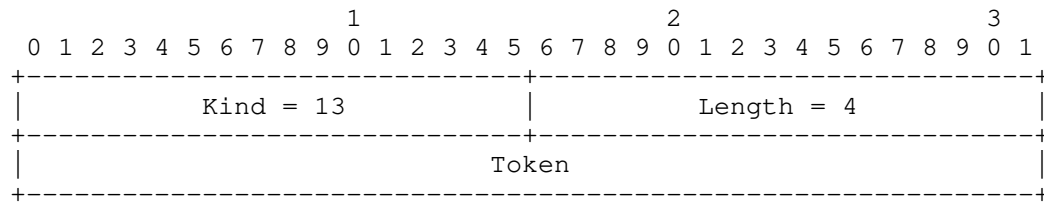


Figure 27: Idempotence Expenditure

- o Kind: 13 (Idempotence Expenditure option)
- o Length: 8
- o Token: The token being spent.

Clients SHOULD prioritize spending the smaller tokens.

The proxy responds by sending either an Idempotence Accepted or Rejected option as part of the Authentication Reply:

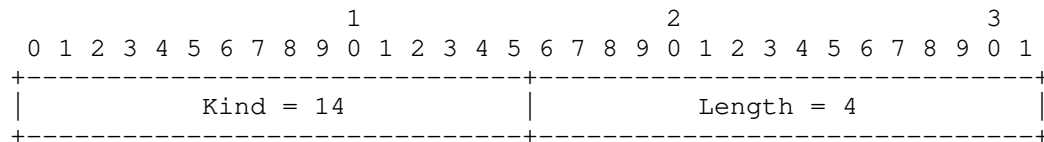


Figure 28: Idempotence Accepted

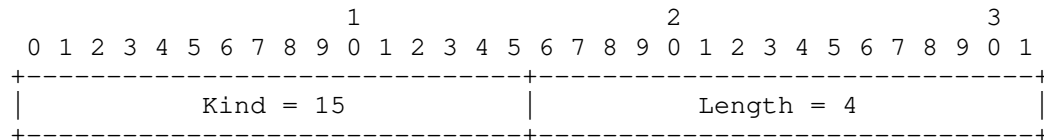


Figure 29: Idempotence Rejected

If eligible, the token is spent before attempting to honor the Request. If the token is not eligible for spending, the Authentication Reply MUST indicate failure.

### 8.5.3. Shifting windows

Windows can be shifted (i. e. have their base increased, while retaining their size) unilaterally by the proxy.

Proxy implementations SHOULD shift the window: \* as soon as the lowest-order token in the window is spent and \* when a sufficiently high-order token is spent.

Proxy implementations SHOULD NOT shift the window's base beyond the highest unspent token.

### 8.5.4. Out-of-order Window Advertisements

Even though the proxy increases the window's base monotonically, there is no mechanism whereby a SOCKS client can receive the Token Window Advertisements in order. As such, clients SHOULD disregard Token Window Advertisements with a Window Base less than the previously known value.

## 9. Username/Password Authentication

Username/Password authentication is carried out as in [RFC1929].

Clients can also attempt to authenticate by placing the Username/Password request in an Authentication Data Option.

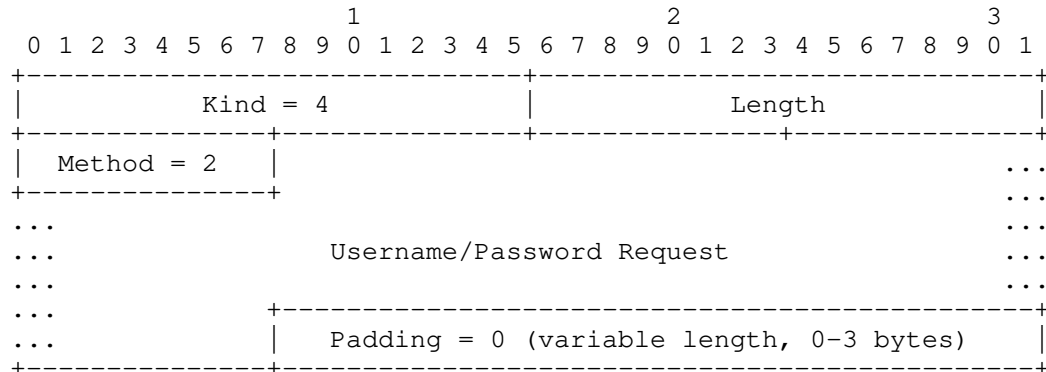


Figure 30: Password authentication via a SOCKS Option

- o Username/Password Request: The Username/Password Request, as described in [RFC1929].



Proxies reply by including a Authentication Data Option in the next Authentication Reply which contains the Username/Password reply:

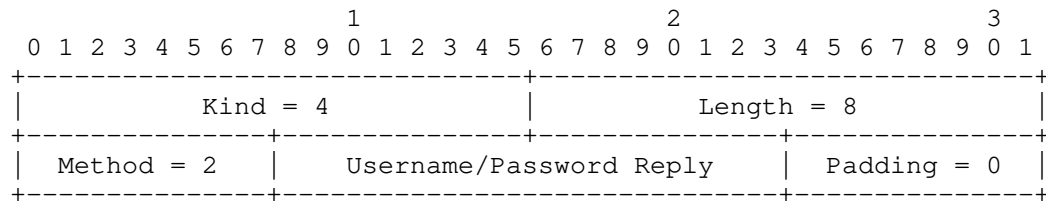


Figure 31: Reply to password authentication via a SOCKS Option

- o Username/Password Reply: The Username/Password Reply, as described in [RFC1929].

#### 10. TCP Fast Open on the Client-Proxy Leg

TFO breaks TCP semantics, causing replays of the data in the SYN's payload under certain rare circumstances [RFC7413]. A replayed SOCKS Request could itself result in a replayed connection on behalf of the client.

As such, client implementations SHOULD NOT use TFO on the client-proxy leg unless:

- o The protocol running on top of SOCKS tolerates the risks of TFO, or
- o The SYN's payload does not contain any application data (so that no data is replayed to the server, even though duplicate connections are still possible), or
- o The client uses Idempotence Options, making replays very unlikely, or
- o SOCKS is running on top of TLS and Early Data is not used.

#### 11. False Starts

In case of CONNECT Requests, the client MAY start sending application data as soon as possible, as long as doing so does not incur the risk of breaking the SOCKS protocol.

Clients must work around the authentication phase by doing any of the following:

- o If the Request does not contain an Authentication Method Advertisement option, the authentication phase is guaranteed not to happen. In this case, application data MAY be sent immediately after the Request.
- o Application data MAY be sent immediately after receiving an Authentication Reply indicating success.
- o When performing a method-specific authentication sequence, application data MAY be sent immediately after the last client message.

## 12. DNS provided by SOCKS

Clients may require information typically obtained from DNS servers, albeit from the proxy's vantage point.

While the CONNECT command can work with domain names, some clients' workflows require that addresses be resolved as a separate step prior to connecting. Moreover, the SOCKS Datagram Header, as described in Section 7.3, can be reduced in size by providing the resolved destination IP address, rather than the FQDN.

Emerging techniques may also make use of DNS to deliver server-specific information to clients. For example, Encrypted SNI [I-D.ietf-tls-esni] relies on DNS to publish encryption keys.

Proxy implementations MAY provide a default plaintext DNS service. A client looking to make use of it issues a CONNECT Request to IP address 0.0.0.0 or 0:0:0:0:0:0:0:0 on port 53. Following successful authentication, the Operation Reply indicates an unspecified bind address (0.0.0.0 or ::) and port (0). The client and proxy then behave as per [RFC7766].

The service itself can be provided directly by the proxy daemon, or by proxying the client's request to a pre-configured DNS server.

If the proxy does not implement such functionality, it MAY return an error code signalling "Connection refused".

## 13. Security Considerations

### 13.1. Large requests

Given the format of the request message, a malicious client could craft a request that is in excess of 16 KB and proxies could be prone to DDoS attacks.

To mitigate such attacks, proxy implementations SHOULD be able to incrementally parse the requests. Proxies MAY close the connection to the client if:

- o the request is not fully received after a certain timeout, or
- o the number of options or their size exceeds an imposed hard cap.

### 13.2. Replay attacks

In TLS 1.3, early data (which is likely to contain a full SOCKS request) is prone to replay attacks.

While Token Expenditure options can be used to mitigate replay attacks, anything prior to the initial Token Request is still vulnerable. As such, client implementations SHOULD NOT make use of TLS early data unless the Request attempts to spend a token.

### 13.3. Resource exhaustion

Malicious clients can issue a large number of Session Requests, forcing the proxy to keep large amounts of state.

To mitigate this, the proxy MAY implement policies restricting the number of concurrent sessions on a per-IP or per-user basis, or barring unauthenticated clients from establishing sessions.

## 14. IANA Considerations

This document requests that IANA allocate 2-byte option kinds for SOCKS 6 options. Further, this document requests the following option kinds:

- o Unassigned: 0
- o Stack: 1
- o Authentication Method Advertisement: 2
- o Authentication Method Selection: 3
- o Authentication Data: 4
- o Session Request: 5
- o Session ID: 6
- o Session Untrusted: 7

- o Session OK: 8
- o Session Invalid: 9
- o Session Teardown: 10
- o Idempotence Request: 11
- o Idempotence Window: 12
- o Idempotence Expenditure: 13
- o Idempotence Accepted: 14
- o Idempotence Rejected: 15
- o Resolution Request: 16
- o IPv4 Resolution: 17
- o IPv6 Resolution: 18
- o Vendor-specific: 64512-0xFFFF

This document also requests that IANA allocate a TCP and UDP port for SOCKS over TLS and DTLS, respectively.

## 15. Acknowledgements

The protocol described in this draft builds upon and is a direct continuation of SOCKS 5 [RFC1928].

## 16. References

### 16.1. Normative References

- [RFC1929] Leech, M., "Username/Password Authentication for SOCKS V5", RFC 1929, DOI 10.17487/RFC1929, March 1996, <<https://www.rfc-editor.org/info/rfc1929>>.
- [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>>.

- [RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <<https://www.rfc-editor.org/info/rfc7766>>.
- [RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <<https://www.rfc-editor.org/info/rfc8305>>.

## 16.2. Informative References

- [I-D.ietf-tls-dtls-connection-id]  
Rescorla, E., Tschofenig, H., and T. Fossati, "Connection Identifiers for DTLS 1.2", draft-ietf-tls-dtls-connection-id-07 (work in progress), October 2019.
- [I-D.ietf-tls-esni]  
Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "Encrypted Server Name Indication for TLS 1.3", draft-ietf-tls-esni-04 (work in progress), July 2019.
- [RFC1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996, <<https://www.rfc-editor.org/info/rfc1928>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <<https://www.rfc-editor.org/info/rfc6824>>.
- [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <<https://www.rfc-editor.org/info/rfc7413>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

## Authors' Addresses

Vladimir Olteanu  
University Politehnica of Bucharest  
313 Splaiul Independentei, Sector 6  
Bucharest  
Romania

Email: vladimir.olteanu@cs.pub.ro

Dragos Niculescu  
University Politehnica of Bucharest  
313 Splaiul Independentei, Sector 6  
Bucharest  
Romania

Email: dragos.niculescu@cs.pub.ro