SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2020

R. Bonica
S. Hegde
Juniper Networks
Y. Kamite
NTT Communications Corporation
A. Alston
D. Henriques
Liquid Telecom
J. Halpern
Ericsson
J. Linkova
Google
G. Chen
Baidu

July 8, 2019

IPv6 Support for Segment Routing: SRv6+ draft-bonica-spring-srv6-plus-04

Abstract

This document describes SRv6+. SRv6+ is a Segment Routing (SR) solution that leverages IPv6. It supports a wide variety of use-cases while remaining in strict compliance with IPv6 specifications. SRv6+ is optimized for for ASIC-based forwarding devices that operate at high data rates.

Table of Contents

1. Overview

Network operators deploy Segment Routing (SR) [RFC8402] so that they can forward packets through SR paths. An SR path provides unidirectional connectivity from its ingress node to its egress node. While an SR path can follow the least cost path from ingress to egress, it can also follow any other path.

An SR path contains one or more segments. A segment provides unidirectional connectivity from its ingress node to its egress node. It also includes a topological instruction that controls its behavior.

The topological instruction is executed on the segment ingress node. It determines the segment egress node and the method by which the segment ingress node forwards packets to the segment egress node.

Per-segment service instructions can augment a segment. Per-segment service instructions, if present, are executed on the segment egress node.

Likewise, a per-path service instruction can augment a path. The per-path service instruction, if present, is executed on the path egress node. Section 3 of this document illustrates the relationship between SR paths, segments and instructions.

A Segment Identifier (SID) identifies each segment. Because there is a one-to-one mapping between segments and the topological instructions that control them, the SID that identifies a segment also identifies the topological instruction that controls it.

A SID is different from the topological instruction that it identifies. While a SID identifies a topological instruction, it does not contain the topological instruction that it identifies. Therefore, a SID can be encoded in relatively few bits, while the topological instruction that it identifies may require many more bits for encoding.

An SR path can be represented by its ingress node as an ordered sequence of SIDs. In order to forward a packet through an SR path, the SR ingress node encodes the SR path into the packet as an ordered sequence of SIDs. It can also augment the packet with service instructions.

Because the SR ingress node is also the first segment ingress node, it executes the topological instruction associated with the first segment. This causes the packet to be forwarded to the first segment egress node. When the first segment egress node receives the packet, it executes any per-segment service instructions that augment the first segment.

If the SR path contains exactly one segment, the first segment egress node is also the path egress node. In this case, that node executes any per-path service instruction that augments the path, and SR forwarding is complete.

If the SR path contains multiple segments, the first segment egress node is also the second segment ingress node. In this case, that node executes the topological instruction associated with the second segment. The above-described procedure continues until the packet arrives at the SR egress node.

In the above-described procedure, only the SR ingress node maintains path information. Segment ingress and egress nodes maintain information regarding the segments in which they participate, but they do not maintain path information.

The SR architecture, described above, can leverage either an MPLS [RFC3031] data plane or an IPv6 [RFC8200] data plane. SR-MPLS [I-D.ietf-spring-segment-routing-mpls] leverages MPLS. SRv6 [I-D.ietf-spring-srv6-network-programming] [I-D.ietf-6man-segment-routing-header] leverages IPv6.

This document describes SRv6+. SRv6+ is another SR variant that leverages IPv6. It supports a wide variety of use-cases while remaining in strict compliance with IPv6 specifications. SRv6+ is optimized for ASIC-based forwarding devices that operate at high data rates. Section 9 of this document highlights differences between SRv6 and SRv6+.

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. Paths, Segments And Instructions

An SRv6+ path is determined by the segments that it contains. It can be represented by its ingress node as an ordered sequence of SIDs.

A segment is determined by its ingress node and by the topological instruction that controls its behavior. The topological instruction determines the segment egress node and the method by which the segment ingress node forwards packets to the segment egress node.

Per-segment service instructions augment, but do not determine, segments. A segment ingress node can:

- Send one packet through a segment with one per-segment service instruction.
- Send another packet through the same segment with a different per-segment service instruction.
- Send another packet through the same segment without any per-segment service instructions.

Likewise, per-path service instructions augment, but do not determine, paths.

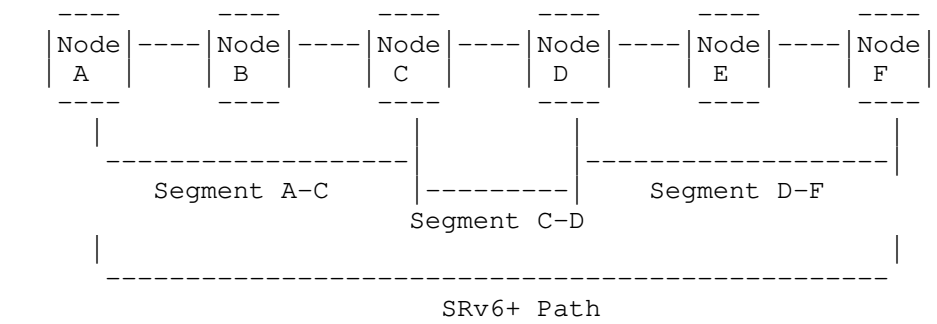

Figure 1: Paths, Segments And Instructions

Figure 1 depicts an SRv6+ path. The path provides unidirectional connectivity from its ingress node (i.e., Node A) to its egress node (i.e., Node F). It contains Segment A-C, Segment C-D and Segment D-F.

In Segment A-C, Node A is the ingress node, Node B is a transit node, and Node C is the egress node. Therefore, the topological instruction that controls the segment is executed on Node A, while

per-segment service instructions that augment the segment (if any exist) are executed on Node C.

In Segment C-D, Node C is the ingress node and Node D is the egress node. Therefore, the topological instruction that controls the segment is executed on Node C, while per-segment service instructions that augment the segment (if any exist) are executed on Node D.

In Segment D-F, Node D is the ingress node, Node E is a transit node, and Node F is the egress node. Therefore, the topological instruction that controls the segment is executed on Node D, while per-segment service instructions that augment the segment (if any exist) are executed on Node F.

Node F is also the path egress node. Therefore, if a per-path service instruction augments the path, it is executed on Node F.

Segments A-C, C-D and D-F are also contained by other paths that are not included in the figure.

4. Segment Types

SRv6+ supports the following segment types:

- strictly routed
- loosely routed

Strictly routed segments forward packets through a specified link that connects the segment ingress node to the segment egress node. Loosely routed segments forward packets through the least cost path from the segment ingress node to the segment egress node.

Each segment type is described below.

4.1. Strictly Routed

When a packet is submitted to a strictly routed segment, the topological instruction associated with that segment operates upon the packet. The topological instruction executes on the segment ingress node and accepts the following parameters:

- An IPv6 address that identifies an interface on the segment egress node.
- A primary interface identifier.
- Zero or more secondary interface identifiers.

The topological instruction behaves as follows:

- If none of the interfaces identified by the above-mentioned parameters are operational, discard the packet and send an ICMPv6 [RFC4443] Destination Unreachable message (Code: 5, Source Route Failed) to the packet's source node.

- Overwrite the packet's Destination Address with the IPv6 address that was received as a parameter.

- If the primary interface is active, forward the packet through the primary interface.

- If the primary interface is not active and any of the secondary interfaces are active, forward the packet through one of the secondary interfaces. Execute procedures so that all packets belonging to a flow are forwarded through the same secondary interface.

4.2. Loosely Routed

When a packet is submitted to a loosely routed segment, the topological instruction associated with that segment operates upon the packet. The topological instruction executes on the segment ingress node and accepts an IPv6 address as a parameter. The IPv6 address identifies an interface on the segment egress node.

The topological instruction behaves as follows:

- If the segment ingress node does not have a viable route to the IPv6 address included as a parameter, discard the packet and send an ICMPv6 Destination Unreachable message (Code:1 Net Unreachable) to the packet's source node.

- Overwrite the packet's Destination Address with the destination address that was included as a parameter.

- Forward the packet to the next hop along the least cost path to the segment egress node. If there are multiple least cost paths to the segment egress node (i.e., Equal Cost Multipath), execute procedures so that all packets belonging to a flow are forwarded through the same next hop.

5. Segment Identifiers (SID)

A Segment Identifier (SID) is an unsigned integer that identifies a segment. Because there is a one-to-one mapping between segments and the topological instructions that control them, the SID that

identifies a segment also identifies the topological instruction that controls it.

A SID is different from the topological instruction that it identifies. While a SID identifies a topological instruction, it does not contain the topological instruction that it identifies. Therefore, a SID can be encoded in relatively few bits, while the topological instruction that it identifies may require many more bits for encoding.

SIDs have node-local significance. This means that a segment ingress node MUST identify each segment that it originates with a unique SID. However, a SID that is used by one segment ingress node to identify a segment that it originates can be used by another segment ingress node to identify another segment.

Although SIDs have node-local significance, an SRv6+ path can be uniquely identified by its ingress node and an ordered sequence of SIDs. This is because the topological instruction associated with each segment determines the ingress node of the next segment (i.e., the node upon which the next SID has significance.)

Although SIDs have node-local significance, they can be assigned in a manner that facilitates debugging. See Section 5.2 and Section 5.3 for details.

5.1. Range

SID values range from 0 to a configurable Maximum SID Value (MSV). The values 0 through 15 are reserved for future use. The following are valid MSVs:

- 65,535 (i.e., 2**16 minus 1)
- 4,294,967,295 (i.e., 2**32 minus 1)

In order to optimize packet encoding (Section 7.1), network operators can configure all nodes within an SRv6+ domain to have the smallest feasible MSV. The following paragraphs explain how an operator determines the smallest feasible MSV.

Consider an SRv6+ domain that contains 5,000 nodes connected to one another by point-to-point infrastructure links. The network topology is not a full-mesh. In fact, each node supports 200 point-to-point infrastructure links or fewer. Given this SRv6+ domain, we will determine the smallest feasible MSV under the following conditions:

- The SRv6+ domain contains strictly routed segments only.

- The SRv6+ domain contains loosely routed segments only.
- The SRv6+ domain contains both strictly and loosely routed segments.

If an SRv6+ domain contains strictly routed segments only, and each node creates a strictly routed segment to each of its neighbors, each node will create 200 segments or fewer and consume 200 SIDs or fewer. This is because each node has 200 neighbors or fewer. Because SIDs have node-local significance (i.e., they can be reused across nodes), the smallest feasible MSV is 65,535.

Adding nodes to this SRv6+ domain will not increase the smallest feasible MSV, so long as each node continues to support 65,519 point-to-point infrastructure links or fewer. If a single node is added to the domain and that node supports 240 infrastructure links, the smallest feasible MSV will increase to 65,535.

If an SRv6+ domain contains loosely routed segments only, and every node creates a loosely routed segment to every other node, every node will create 4,999 segments and consume 4,999 SIDs. This is because the domain contains 5,000 nodes. Because SIDs have node-local significance (i.e., they can be reused across nodes), the smallest feasible MSV is 65,535.

Adding nodes to this SRv6+ domain will not increase the smallest feasible MSV until the number of nodes exceeds 65,519. When the smallest feasible MSV increases, it becomes 4,294,967,295.

If an SRv6+ domain contains both strictly and loosely routed segments, each node will create 5,199 segments or fewer and consume 5,199 SIDs or fewer. This value is the sum of the following:

- The number of loosely routed segments that each node will create, given that every node creates a loosely routed segment to every other node (i.e., 4,999).
- The number of strictly routed segments that each node will create, given that each node creates a strictly routed segment to each of its neighbors (i.e., 200 or fewer).

Because SIDs have node-local significance (i.e., they can be reused across nodes), the smallest feasible MSV is 65,535.

Adding nodes to this SRv6+ domain will not increase the smallest feasible MSV until the number of nodes plus the maximum number of infrastructure links per node exceeds 65,519. When the smallest feasible MSV increases, it becomes 4,294,967,295.

5.2. Assigning SIDs to Strictly Routed Segments

Network operators can establish conventions by which they assign SIDs to strictly routed segments. These conventions can facilitate debugging.

For example, a network operator can reserved a range of SIDs for strictly routed segments. It can further divide that range into subranges, so that all segments sharing a common egress node are identified by SIDs from the same subrange.

5.3. Assigning SIDs to Loosely Routed Segments

In order to facilitate debugging, all loosely routed segments that share a common egress node are identified by the same SID. In order to maintain this discipline, network wide co-ordination is required.

For example, assume that an SRv6+ domain contains N nodes. Network administrators reserve a block of N SIDs and configure one of those SIDs on each node. Each node advertises its SID into the control plane. When another node receives that advertisement, it creates a loosely routed segment between itself and the advertising node. It also associates the SID that it received in the advertisement with the newly created segment. See [I-D.bonica-lsr-crh-isis-extensions] for details.

6. Service Instructions

SRv6+ supports the following service instruction types:

- Per-segment
- Per-path

Each is described below.

6.1. Per-Segment

Per-segment service instructions can augment a segment. Per-segment service instructions, if present, are executed on the segment egress node. Because the path egress node is also a segment egress node, it can execute per-segment service instructions.

The following are examples of per-segment service instructions:

- Expose a packet to a firewall policy.
- Expose a packet to a sampling policy.

Per-segment Service Instruction Identifiers identify a set of service instructions. Per-segment Service Instruction Identifiers are allocated and distributed by a controller. They have domain-wide significance.

6.2. Per-Path

A per-path service instruction can augment a path. The per-path service instruction, if present, is executed on the path egress node.

The following are examples of per-path service instructions:

- De-encapsulate a packet and forward its newly exposed payload through a specified interface.
- De-encapsulate a packet and forward its newly exposed payload using a specified routing table.

Per-path Service Instruction Identifiers identify per-path service instructions. Per-path Service Instruction Identifiers are allocated and distributed by the processing node (i.e., the path egress node). They have node-local significance. This means that the path egress node MUST allocate a unique Per-path Service Instruction Identifier for each per-path service instruction that it instantiates.

7. The IPv6 Data Plane

SRv6+ ingress nodes generate IPv6 header chains that represent SRv6+ paths. An IPv6 header chain contains an IPv6 header. It can also contain one or more extension headers.

An extension header chain that represents an SRv6+ path can contain any valid combination of IPv6 extension headers. The following bullet points describe how SRv6+ leverages IPv6 extension headers:

- If an SRv6+ path contains multiple segments, the IPv6 header chain that represents it MUST contain a Routing header. The SRv6+ path MUST be encoded in the Routing header as an ordered sequence of SIDs.
- If an SRv6+ path is augmented by a per-path service instruction, the IPv6 header chain that represents it MUST contain a Destination Options header. The Destination Options header MUST immediately precede an upper-layer header and it MUST include a Per-Path Service Instruction Identifier.
- If an SRv6+ path contains a segment that is augmented by a per-segment service instruction, the IPv6 chain that represents it

MUST contain a Routing header and a Destination Options header. The Destination Options header MUST immediately precede a Routing header and it MUST include the Per-Segment Service Instruction Identifier.

The following subsections describe how SRv6+ uses the Routing header and the Destination Options header.

7.1. The Routing Header

SRv6+ defines a new Routing header type, called the Compressed Routing Header (CRH) [I-D.bonica-6man-comp-rtg-hdr]. The CRH contains the following fields:

- Next Header - Identifies the header immediately following the CRH.
- Hdr Ext Len - Length of the CRH.
- Routing Type - Identifies the Routing header variant (i.e., CRH)
- Segments Left - The number of segments still to be traversed before reaching the path egress node.
- Last Entry - Represents the index of the last element of the Segment List.
- Com (Compression) - Represents the length of each entry in the SID List. Values are reserved (0), sixteen bits (1), thirty-two bits (2), and reserved (3). In order to maximize header compression, this value should reflect the smallest feasible MSV (Section 5.1).
- SID List - Represents the SRv6+ path as an ordered list of SIDs. SIDs are listed in reverse order, with SID[0] representing the final segment, SID[1] representing the penultimate segment, and so forth. SIDs are listed in reverse order so that Segments Left can be used as an index to the SID List. The SID indexed by Segments Left is called the current SID.

As per [RFC8200], when an IPv6 node receives a packet, it examines the packet's destination address. If the destination address represents an interface belonging to the node, the node processes the next header. If the node encounters and recognizes the CRH, it processes the CRH as follows:

- If Segments Left equal 0, skip over the CRH and process the next header in the packet.
- Decrement Segments Left.

- Search for the current SID in a local table that maps SID's to topological instructions. If the current SID cannot be found in that table, send an ICMPv6 Parameter Problem message to the packet's Source Address and discard the packet.

- Execute the topological instruction found in the table as described in Section 4. This causes the packet to be forwarded to the segment egress node.

When the packet arrives at the segment egress node, the above-described procedure is repeated.

7.2. The Destination Options Header

According to [RFC8200], the Destination Options header contains one or more IPv6 options. It can occur twice within a packet, once before a Routing header and once before an upper-layer header. The Destination Options header that occurs before a Routing header is processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations that are listed in the Routing header. The Destination Options header that occurs before an upper-layer header is processed by the packet's final destination only.

Therefore, SRv6+ defines the following new IPv6 options:

- The SRv6+ Per-Segment Service Instruction Option [I-D.bonica-6man-seg-end-opt]

- The SRv6+ Per-Path Service Instruction Option [I-D.bonica-6man-vpn-dest-opt]

The SRv6+ Per-Segment Service Instruction Option is encoded in a Destination Options header that precedes the CRH. Therefore, it is processed by every segment egress node. It includes a Per-Segment Service Instruction Identifier and causes segment egress nodes to execute per-segment service instructions.

The SRv6+ Per-Path Service Instruction Option is encoded in a Destination Options header that precedes the upper-layer header. Therefore, it is processed by the path egress node only. It includes a Per-Path Service Instruction Identifier and causes the path egress node to execute a per-path service instruction.

8. Control Plane

IS-IS extensions [I-D.bonica-lsr-crh-isis-extensions] have been defined for the following purposes:

- So that SRv6+ segment ingress nodes can flood information regarding strictly routed segments that they originate
- So that SRv6+ segment egress nodes can flood information regarding loosely routed segments that they terminate

BGP extensions [I-D.ssangli-idr-bgp-vpn-srv6-plus] are being defined so that SRv6+ path egress nodes can associated path-terminating service instructions with Network Layer Reachability Information (NLRI).

9. Differences Between SRv6 and SRv6+

9.1. Routing Header Size

SRv6 defines a Routing header type, called the Segment Routing Header (SRH). The SRH contains a field that represents the SRv6 path as an ordered sequence of SIDs. Each SID contained by that field is 128 bits long.

Likewise, SRv6+ defines a Routing Header Type, called the Compressed Routing Header (CRH). The CRH contains a field that represents the SRv6+ path as an ordered sequence of SIDs. Within that field, SIDs can be 16 or 32 bits long.

SIDs	SRv6 SRH (128-bit SID)	SRv6+ CRH (16-bit SID)	SRv6+ CRH (32-bit SID)
1	24	16	16
2	40	16	16
3	56	16	24
4	72	16	24
5	88	24	32
6	104	24	32
7	120	24	40
8	136	24	40
9	152	32	48
10	168	32	48
11	184	32	56
12	200	32	56
13	216	40	64
14	232	40	64
15	248	40	72
16	264	40	72

Table 1: Routing Header Size (in Bytes) As A Function Of Routing Header Type and Number Of SIDs

Table 1 reflects Routing header size as a function of Routing header type and Number of SIDs contained by the Routing header.

Large Routing headers are undesirable for the following reasons:

- Many ASIC-based forwarders copy the entire IPv6 extension header chain from buffer memory to on-chip memory. As the size of the IPv6 extension header chain increases, so does the cost of this copy.

- Because Path MTU Discovery (PMTUD) [RFC8201] is not entirely reliable, many IPv6 hosts refrain from sending packets larger than the IPv6 minimum link MTU (i.e., 1280 bytes). When packets are small, the overhead imposed by large Routing headers becomes pronounced.

9.2. Decoupling of Topological and Service Instructions

SRv6+ decouples topological instructions from service instructions. Topological instructions are invoked at the segment ingress node, as a result of CRH processing, while service instructions are invoked at the segment egress node, as a result of Destination Option

processing. Therefore, network operators can use SRv6+ mechanisms to support topological instructions, service instructions, or both.

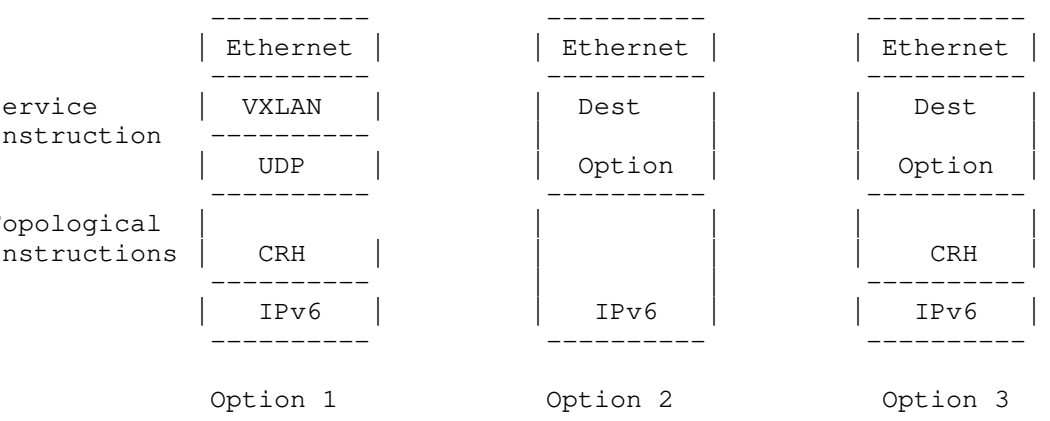

Figure 2: EVPN Design Alternatives

Figure 2 illustrates this point by depicting design options available to network operators offering Ethernet Virtual Private Network [RFC7432] services over Virtual eXtensible Local Area Network (VXLAN) [RFC7348]. In Option 1, the network operator encodes topological instructions in the CRH, while encoding service instructions in a VXLAN header. In Option 2, the network operator encodes service instructions in a Destination Options header, while allowing traffic to traverse the least cost path between the ingress and egress Provider Edge (PE) routers. In Option 3, the network operator encodes topological instructions in the CRH, and encodes service instructions in a Destination Options header.

9.3. Authentication

The IPv6 Authentication Header (AH) [RFC4302] can be used to authenticate SRv6+ packets. However, AH processing is not defined in SRv6.

9.4. Traffic Engineering Capability

SRv6+ supports traffic engineering solutions that rely exclusively upon strictly routed segments. For example, consider an SRv6+ network whose diameter is 12 hops and whose minimum feasible MSV is 65,525. In that network, in the worst case, SRv6+ overhead is 72 bytes (i.e., a 40-byte IPv6 header and a 32-byte CRH).

SRv6 also supports traffic engineering solutions that rely exclusively upon strictly routed segments (i.e., END.X SIDs).

However, SRv6 overhead may be prohibitive. For example, consider an SRv6 network whose diameter is 12 hops. In the worst case, SRv6 overhead is 240 bytes (i.e., a 40 byte IPv6 header and a 200-byte SRH).

9.5. IP Addressing Architecture

In SRv6, an IPv6 address can represent either of the following:

- A network interface
- An instruction instantiated on a node (i.e., an SRv6 SID)

In SRv6+ an IPv6 address always represents a network interface, as per [RFC4291].

10. Compliance

In order to be compliant with this specification, an SRv6+ implementation MUST:

- Be able to process IPv6 options as described in Section 4.2 of [RFC8200].
- Be able to process the Routing header as described in Section 4.4 of [RFC8200].
- Be able to process the Destination Options header as described in Section 4.6 of [RFC8200].
- Recognize the CRH.
- Be able to encode an SRv6+ path in the CRH as an ordered sequence of 32-bit SIDs.
- Be able to process a CRH that includes 32-bit SIDs.

Additionally, an SRv6+ implementation MAY:

- Be able to encode an SRv6+ path in the CRH as an ordered sequence of 16-bit SIDs.
- Be able to process a CRH that includes 16-bit SIDs.
- Recognize the Per-Segment Service Instruction Option.
- Recognize the Per-Path Service Instruction Option.

11. Operational Considerations

11.1. Ping and Traceroute

Ping and Traceroute [RFC2151] both operate correctly in SRv6+ (i.e., in the presence of the CRH).

11.2. ICMPv6 Rate Limitting

As per [RFC4443], SRv6+ nodes rate limit the ICMPv6 messages that they emit.

11.3. SID Lengths And SID Length Transitions

An SRv6+ implementation MAY include a configuration option that determines how it encodes SIDs (i.e., in 16 or 32 bits). In order to reduce operational complexity, network operators typically configure their networks so that every node encodes SIDs identically.

As a network grows, its minimum feasible MSV may increase. In this case, the network may need to migrate from one SID encoding to another. The following bullet points describe a migration strategy for an SRv6+ network that is migrating from 16-bit SIDs to 32-bit SIDs:.

- Ensure that all nodes can process a CRH that includes 32-bit SIDs.
- Configure each nodes so that encodes SIDs in 32-bits.
- Configure SIDs whose value exceeds 65,535.

12. IANA Considerations

SID values 0-15 are reserved for future use. They may be assigned by IANA, based on IETF Consensus.

IANA is requested to establish a "Registry of SRv6+ Reserved SIDs". Values 0-15 are reserved for future use.

13. Security Considerations

SRv6+ domains MUST NOT span security domains. In order to enforce this requirement, security domain edge routers MUST do one of the following:

- Discard all inbound SRv6+ packets
- Authenticate [RFC4302] [RFC4303] all inbound SRv6+ packets

14. Acknowledgements

The authors wish to acknowledge Dr. Vanessa Ameen and John Scudder.

15. References

15.1. Normative References

[I-D.bonica-6man-comp-rtg-hdr]
Bonica, R., Kamite, Y., Niwa, T., Alston, A., Henriques, D., So, N., Xu, F., Chen, G., Zhu, Y., Yang, G., and Y. Zhou, "The IPv6 Compressed Routing Header (CRH)", draft-bonica-6man-comp-rtg-hdr-04 (work in progress), May 2019.

[I-D.bonica-6man-seg-end-opt]
Bonica, R., Halpern, J., So, N., Xu, F., Chen, G., Zhu, Y., Yang, G., and Y. Zhou, "The IPv6 Segment Endpoint Option", draft-bonica-6man-seg-end-opt-03 (work in progress), March 2019.

[I-D.bonica-6man-vpn-dest-opt]
Bonica, R., Lenart, C., So, N., Xu, F., Presbury, G., Chen, G., Zhu, Y., Yang, G., and Y. Zhou, "The IPv6 Virtual Private Network (VPN) Context Information Option", draft-bonica-6man-vpn-dest-opt-05 (work in progress), March 2019.

[I-D.bonica-lsr-crh-isis-extensions]
Kaneriya, P., Shetty, R., Hegde, S., and R. Bonica, "IS-IS Extensions To Support The IPv6 Compressed Routing Header (CRH)", draft-bonica-lsr-crh-isis-extensions-00 (work in progress), May 2019.

[I-D.ssangli-idr-bgp-vpn-srv6-plus]
Ramachandra, S. and R. Bonica, "BGP based Virtual Private Network (VPN) Services over SRv6+ enabled IPv6 networks", draft-ssangli-idr-bgp-vpn-srv6-plus-00 (work in progress), July 2019.

[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>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[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>.

15.2. Informative References

[I-D.ietf-6man-segment-routing-header] Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-21 (work in progress), June 2019.

[I-D.ietf-spring-segment-routing-mpls] Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-22 (work in progress), May 2019.

[I-D.ietf-spring-srv6-network-programming] Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-00 (work in progress), April 2019.

[RFC2151] Kessler, G. and S. Shepard, "A Primer On Internet and TCP/IP Tools and Utilities", FYI 30, RFC 2151, DOI 10.17487/RFC2151, June 1997, <https://www.rfc-editor.org/info/rfc2151>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.

[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>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <https://www.rfc-editor.org/info/rfc8201>.

Authors' Addresses

Ron Bonica
Juniper Networks
Herndon, Virginia 20171
USA

Email: rbonica@juniper.net

Shraddha Hegde
Juniper Networks
Embassy Business Park
Bangalore, KA 560093
India

Email: shraddha@juniper.net

Yuji Kamite
NTT Communications Corporation
3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Email: : y.kamite@ntt.com

Andrew Alston
Liquid Telecom
Nairobi
Kenya

Email: Andrew.Alston@liquidtelecom.com

Daniam Henriques
Liquid Telecom
Johannesburg
South Africa

Email: daniam.henriques@liquidtelecom.com

Joel Halpern
Ericsson
P. O. Box 6049
Leesburg, Virginia 20178
USA

Email: joel.halpern@ericsson.com

Jen Linkova
Google
Mountain View, California 94043
USA

Email: furry@google.com

Gang Chen
Baidu
No.10 Xibeiwang East Road Haidian District
Beijing 100193
P.R. China

Email: phdgang@gmail.com

intarea
Internet-Draft
Intended status: Standards Track
Expires: December 20, 2019

P. Pfister
E. Vyncke, Ed.
Cisco
T. Pauly
Apple
D. Schinazi
Google LLC
W. Shao
Cisco

June 18, 2019

Discovering Provisioning Domain Names and Data draft-ietf-intarea-provisioning-domains-05

Abstract

An increasing number of hosts access the Internet via multiple interfaces or, in IPv6 multi-homed networks, via multiple IPv6 prefix configurations context.

This document describes a way for hosts to identify such contexts, called Provisioning Domains (PvDs), where Fully Qualified Domain Names (FQDNs) act as PvD identifiers. Those identifiers are advertised in a new Router Advertisement (RA) option and, when present, are associated with the set of information included within the RA.

Based on this FQDN, hosts can retrieve additional information about their network access characteristics via an HTTP over TLS query. This allows applications to select which Provisioning Domains to use as well as to provide configuration parameters to the transport layer and above.

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 December 20, 2019.

Table of Contents

1. Introduction

It has become very common in modern networks for hosts to access the internet through different network interfaces, tunnels, or next-hop routers. For example, if Alice has a mobile phone provider and a broadband provider in her home, her devices and her applications should be capable of seamlessly transitioning from one to the other and be able to use her Wi-Fi to access local resources or use the more suitable link on a per-application base. This document provides the basic information necessary to make this choice intelligently. There are similar use cases for IPsec Virtual Private Networks that are already considered Explicit PvDs in [RFC7556].

To describe the set of network configurations associated with each access method, the concept of Provisioning Domain (PvD) was defined in [RFC7556].

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) could 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 identical services.

This document also introduces a way for hosts to retrieve optional and 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 unfit, or 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.

2. Terminology

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 [RFC2119].

In addition, 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. 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.

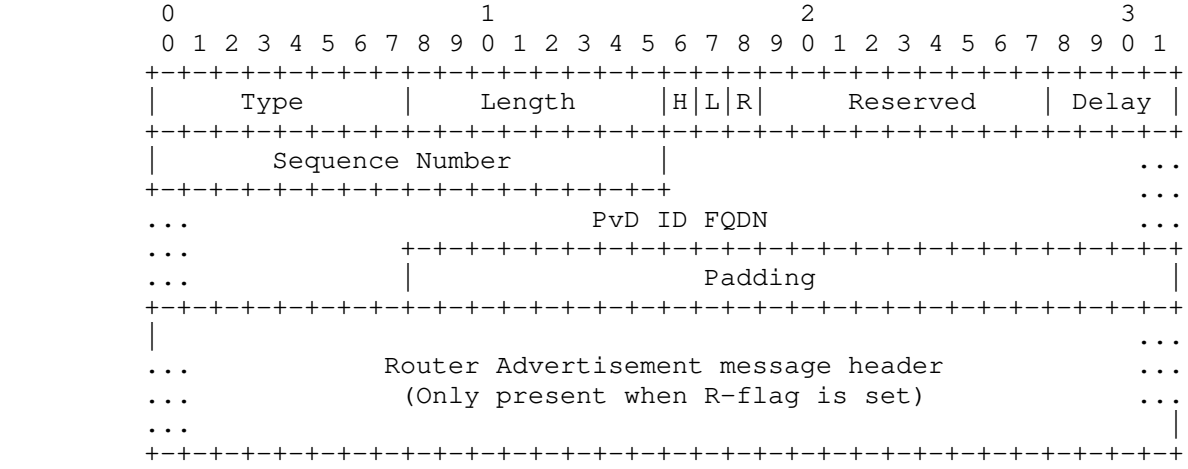

Figure 1: PvD ID Router Advertisements Option format

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]).

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

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

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 octets boundary. 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 such 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 a RDNSS and prefix information options (it also have the sequence number 123, presence of additional information to be fetched with a delay indicated as 5):

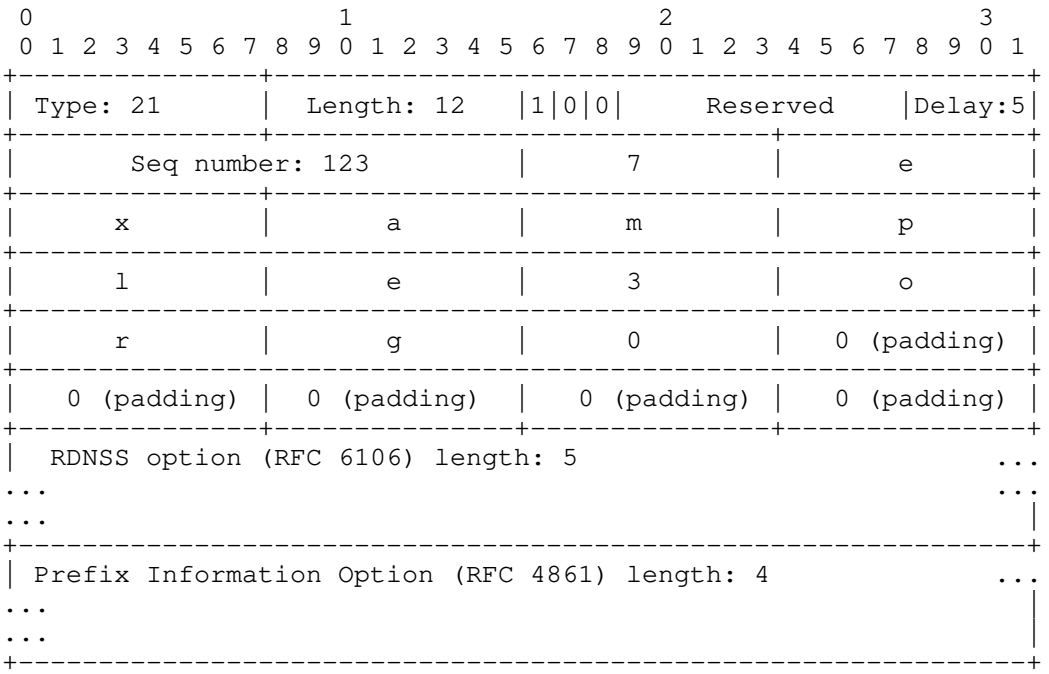

Figure 2

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. In particular, 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 others.

In order to provide multiple different PvDs, a router MUST send multiple RAs. Different explicit PvDs MAY be advertised with RAs using the same IPv6 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. 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 (i.e., A=a), assuming ASCII with zero parity while non-alphabetic codes must match exactly (see also Section 3.1 of [RFC1035]). 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 ([RFC2461], [RFC4191] and [RFC8028]), hosts MAY consider only the configuration associated with an arbitrary set 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 configuration elements using DHCPv6 (e.g., addresses or DNS recursive resolvers), they MUST be associated with the explicit or implicit PvD of the RA received on the same interface, sent from the same LLA, and with the O-flag or M-flag set [RFC4861]. If no such PvD is found, or whenever multiple different PvDs are found, the host behavior is unspecified.

This process requires hosts to keep track of received RAs, associated PvD IDs, and routers LLA; it also assumes that the router either acts as a DHCPv6 server or relay and uses the same LLA for DHCPv6 and RA traffic (which may not be the case when the router uses VRRP to send its RA).

3.4.2. DHCPv4 configuration association

When a host retrieves configuration elements from DHCPv4, they MUST be associated with the explicit PvD received on the same interface, whose PVD Options L-flag is set and, in the case of a non point-to-point link, using the same datalink address. If no such PvD is found, or whenever multiple different PvDs are found, the configuration elements coming from DHCPv4 MUST be associated with the implicit PvD identified by the interface on which the DHCPv4 transaction happened. The case of multiple explicit PvD for an IPv4 interface is undefined.

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:

The same PVD-ID FQDN.

The same H-bit, Delay and Sequence Number values.

The L bit set whenever the host is sharing IPv4 connectivity received from the same upstream interface.

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 strongly associated with the corresponding PvD. Specificially, queries sent to a configured recursive DNS server SHOULD be sent from a local IP address that belongs to the matching PvD. Answers received from the DNS server SHOULD only be used on the same PvD.

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

A PvD associated with a VPN or otherwise private network may provide DNS answers that contain addresses inaccessible over another PvD.

A PvD that uses a NAT64 [RFC6146] and DNS64 [RFC6147] will synthesize IPv6 addresses in DNS answers that are not globally routable, and cannot be used on other PvDs. Conversely, an IPv4 address resolved via DNS on another PvD cannot be directly used on a NAT64 network without the host synthesizing an IPv6 address.

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 [RFC7159].

The purpose of this additional set of information is to securely provide additional information to applications about the connectivity that is provided using a given interface and source address pair. 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.

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 [RFC5785]. 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 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. When a JSON object could not be retrieved, an error message SHOULD be logged and/or displayed in a rate-limited fashion.

After retrieval of the PvD Additional Information, hosts MUST keep track of the Sequence Number value received in subsequent RAs including the same PvD ID. In case the new value is greater than the value that was observed when the PvD Additional Information object was retrieved (using serial number arithmetic comparisons [RFC1982]), or whenever the validity time included in the PVD Additional Information JSON object is expired, hosts MUST either perform a new query and retrieve a new version of the object, or, failing that, deprecate the object and stop using the additional information provided in the JSON object.

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.

When a host performs an object update after it detected a change in the PvD Option Sequence number, it MUST delay the query by a random time between zero and 2**(Delay * 2) milliseconds, where 'Delay' corresponds to the 4 bits long unsigned integer in the last received PvD Option.

When a host last retrieved an object at time A including a validity time B, and is configured to keep the object up to date, it MUST perform the update at 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 MUST delay the query by 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 the PIO is not covered by at least one of the listed prefixes, the PvD associated with the tested prefix MUST be considered unsafe and MUST NOT be used. While this does not prevent a malicious network provider, it does complicate some attack scenarios, and may help detecting misconfiguration.

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 MAY increment the PVD Option Sequence number in order to inform host that a new PvD Additional Information object is available and should be retrieved.

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
name	Human-readable service name	UTF-8 string [RFC3629]	"Awesome Wi-Fi"
expires	Date after which this object is not valid	[RFC3339]	"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 a valid string associated with the "name" key at the root of the object, or a valid date associated with the "expires" key, also at the root of the object, MUST be ignored. In such cases, an error message SHOULD be logged and/or displayed in a rate-limited fashion. 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
localizedName	Localized user-visible service name, language can be selected based on the HTTP Accept-Language header in the request.	UTF-8 string	"Wi-Fi Genial"
dnsZones	DNS zones searchable and accessible	array of DNS zones	["example.com","sub.example.org"]
noInternet	No Internet, set when the PvD only provides restricted access to a set of services	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 implementers or vendors denomination. Upon receiving such a sub-dictionary, host MUST ignore this sub-dictionary if it is unknown. When the vendor or implementor 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 examples show how the JSON keys defined in this document can be used:

```
{
  "name": "Foo Wireless",
  "localizedName": "Foo-France Wi-Fi",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes" : ["2001:db8:1::/48", "2001:db8:4::/48"],
}

{
  "name": "Bar 4G",
  "localizedName": "Bar US 4G",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
}

{
  "name": "Company Network",
  "localizedName": "Company Network",
  "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 the RA PIO are covered by a prefix from the PvD Additional Information. An adversarial router willing to fake the use of a given explicit PvD, without any access to the actual PvD Additional Information, would need to perform NAT66 in order to circumvent this check.

It is also RECOMMENDED that the HTTPS server checks the IPv6 source addresses of incoming connections (see Section 4.1). This check give reasonable assurance that neither NPTv6 [RFC6296] nor NAT66 were used and restricts the information to the valid network users.

Note that this check cannot be performed when the HTTPS query is performed over IPv4. Therefore, the PvD ID FQDN SHOULD NOT have a DNS A record whenever all hosts using the given PvD have IPv6 connectivity.

5. Operational Considerations

This section describes some 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. For example, a RA message containing some options and a PvD option that also contains other options will be described as:

- RA Header: router lifetime = 6000
- Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- 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

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. For example, here is the RA for non-PvD-aware hosts:

- RA Header: router lifetime = 6000 (non-PvD-aware hosts will use this router as a default router)
- Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- Recursive DNS Server Option: length = 3, addresses= [2001:db8:cafe::53]
- 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 a RA example for PvD-aware hosts:

- RA Header: router lifetime = 0 (non-PvD-aware hosts will not use this router as a default router)

- PvD Option header: length = 3 + 2 + 4 + 3, PvD ID FQDN = 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 adress.

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 advertize wrong information that could lead applications or hosts to select an hostile PvD. Users should always apply caution when connecting to an unknown network.

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).

IANA is asked to assign the value "pvd" from the Well-Known URIs registry.

8.1. 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 [RFC8126].

8.2. 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 [RFC8126], via a Standards Track RFC document.

8.3. 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. Acknowledgements

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 Chow, Lorenzo Colitti, Michael Di Bartolomeo, Ian Farrer, Phillip Hallam-Baker, Bob Hinden, Tatuya Jinmei, Erik Kline, Ted Lemon, Jen Lenkova, Veronika McKillop, Mark Townsley and James Woodyatt for useful and interesting discussions and reviews.

Finally, special thanks to Thierry Danis and Wenqin Shao for their valuable inputs and implementation efforts ([github]), 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>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[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>.

[RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, DOI 10.17487/RFC2461, December 1998, <https://www.rfc-editor.org/info/rfc2461>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[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>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

[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>.

10.2. Informative references

[github] Cisco, "IPv6-mPvD github repository", <https://github.com/IPv6-mPvD>.

[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, "IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std".

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/info/rfc5785>.

[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>.

[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>.

[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>.

[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>.

[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] IANA, "URN Namespaces", <https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml#urn-namespaces-1>.

Appendix A. Changelog

Note to RFC Editors: Remove this section before publication.

A.1. Version 00

Initial version of the draft. Edited by Basile Bruneau + Eric Vyncke and based on Basile's work.

A.2. Version 01

Major rewrite intended to focus on the the retained solution based on corridors, online, and WG discussions. Edited by Pierre Pfister. The following list only includes major changes.

PvD ID is an FQDN retrieved using a single RA option. This option contains a sequence number for push-based updates, a new H-flag, and a L-flag in order to link the PvD with the IPv4 DHCP server.

A lifetime is included in the PvD ID option.

Detailed Hosts and Routers specifications.

Additional Information is retrieved using HTTP-over-TLS when the PvD ID Option H-flag is set. Retrieving the object is optional.

The PvD Additional Information object includes a validity date.

DNS-based approach is removed as well as the DNS-based encoding of the PvD Additional Information.

Major cut in the list of proposed JSON keys. This document may be extended later if need be.

Monetary discussion is moved to the appendix.

Clarification about the 'prefixes' contained in the additional information.

Clarification about the processing of DHCPv6.

A.3. Version 02

The FQDN is now encoded with ASCII format (instead of DNS binary) in the RA option.

The PvD ID option lifetime is removed from the object.

Use well known URI "https://<PvD-ID>/.well-known/pvd"

Reference RFC3339 for JSON timestamp format.

The PvD ID Sequence field has been extended to 16 bits.

Modified host behavior for DHCPv4 and DHCPv6.

Removed IKEv2 section.

Removed mention of RFC7710 Captive Portal option. A new I.D. will be proposed to address the captive portal use case.

A.4. WG Document version 00

Document has been accepted as INTAREA working group document

IANA considerations follow RFC8126 [RFC8126]

PvD ID FQDN is encoded as per RFC 1035 [RFC1035]

PvD ID FQDN is prepended by a one-byte length field

Marcus Keane added as co-author

dnsZones key is added back

draft of a privacy consideration section and added that a temporary address should be used to retrieve the PvD additional information

per Bob Hinden's request: the document is now aiming at standard track and security considerations have been moved to the main section

A.5. WG Document version 01

Removing references to 'metered' and 'characteristics' keys. Those may be in scope of the PvD work, but this document will focus on essential parts only.

Removing appendix section regarding link quality and billing information.

The PvD RA Option may now contain other RA options such that PvD-aware hosts may receive configuration information otherwise invisible to non-PvD-aware hosts.

Clarify that the additional PvD Additional Information is not intended to modify host's networking stack behavior, but rather provide information to the Application, used to select which PvDs must be used and provide configuration parameters to the transport layer.

The RA option padding is used to increase the option size to the next 64 (was 32) bits boundary.

Better detail the Security model and Privacy considerations.

A.6. WG Document version 02

Use the IANA value of 21 in the text and update the IANA considerations section accordingly

add the Delay field to avoid the thundering herd effect

add Wenqin Shao as author

keep the 1 PvD per RA model

changed the intro (per Zhen Cao) "when choosing which PvD and transport should be used" => "when choosing which PvD should be used"

rename A-flag in R-flag to avoid A-flag of PIO

use the wording "PvD Option", removing the ID token as it is now a container with more then just an ID, removing 'RA' in the option name to be consistent with other IANA NDP option

use "non-PvD-aware" rather than "PvD-ignorant"

added more reference to RFC 7556 (notably for PvD being globally unique, introducing PvD-aware host vs. PvD-aware node)

Section 3.4.3 renamed from "interconnection shared by node" to 'connection shared by node"

Section 3.4 renamed into "PvD-aware Host Behavior"

Added a section "Non-PvD-aware Host Behavior"

A.7. WG Document version 04

Updated reference for DHCPv6-PD from RFC 3633 to RFC 8415.

Enhanced IANA considerations to clarify review process and new registries.

Added a section on considerations for handling DNS on a PvD-aware host.

A.7.1. WG Document version 05

Fixed nits about IPSEC and WiFi

Added use case per Phillip Hallam-Baker

Clarified some sentences

Authors' Addresses

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

Email: ppfister@cisco.com

Eric Vyncke (editor)
Cisco
De Kleetlaan, 6
Diegem 1831
Belgium

Email: evyncke@cisco.com

Tommy Pauly
Apple
One Apple Park Way
Cupertino, California 95014
USA

Email: tpauly@apple.com

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043
USA

Email: dschinazi.ietf@gmail.com

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

Email: wenshao@cisco.com

Internet Area Working Group V. Olteanu
Internet-Draft D. Niculescu
Intended status: Experimental University Politehnica of Bucharest
Expires: January 9, 2020 July 08, 2019

SOCKS Protocol Version 6
draft-olteanu-intarea-socks-6-07

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 January 9, 2020.

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

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 satelite.

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:

- 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.

- 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.

- The protocol can be extended via options without breaking backward-compatibility.
- 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-07

- All fields are now alignned.
- Eliminated version minors
- 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
- 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

- Removed requirements w.r.t. caching authentication methods by multihomed clients
- UDP: 8-byte association IDs
- 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.
- Address Resolution options

draft-06

- Session options
- Options now have a 2-byte length field.
- 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
- 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.)

- Shrunk vendor-specific option range to 32 (down from 64).

- Removed reference to dropping initial data. (It could no longer be done as of -05.)

- Initial data size capped at 16KB.

- Application data is never encrypted by SOCKS 6. (It can still be encrypted by the TLS layer under SOCKS.)

- Messages now carry the total length of the options, rather than the number of options. Limited options length to 16KB.

- Security Considerations

    * Updated the section to reflect the smaller maximum message size.

    * Added a subsection on resource exhaustion.

draft-05

- Limited the "slow" authentication negociations to one (and Authentication Replies to 2)

- 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.

- Request: clarified the meaning of the Address and Port fields.

- Better reverse TCP proxy support: optional listen backlog for TCP BIND
- TFO options can no longer be placed inside Operation Replies.
- IP TOS stack option
- Suggested a range for vendor-specific options.
- Revamped UDP functionality
  - Now using fixed UDP ports
  - DTLS support
- Stack options: renamed Proxy-Server leg to Proxy-Remote leg

draft-04

- Moved Token Expenditure Replies to the Authentication Reply.
- Shifted the Initial Data Size field in the Request, in order to make it easier to parse.

draft-03

- Shifted some fields in the Operation Reply to make it easier to parse.
- Added connection attempt timeout response code to Operation Replies.
- Proxies send an additional Authentication Reply after the authentication phase. (Useful for token window advertisements.)
- Renamed the section "Connection Requests" to "Requests"
- Clarified the fact that proxies don’t need to support any command in particular.
- Added the section "TCP Fast Open on the Client-Proxy Leg"
- 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 advetise 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.

- NOOP commands now trigger Operation Replies.
- Renamed Authentication options to Authentication Data options.
- Authentication Data options are no longer mandatory.
- Authentication methods are now advertised via options.
- Shifted some Request fields.
- Option range for vendor-specific options.
- Socket options.
- Password authentication.
- Salt options.

draft-01

- Added this section.
- Support for idempotent commands.
- Removed version numbers from operation replies.
- Request port number for SOCKS over TLS. Deprecate encryption/ encapsulation within SOCKS.
- Added Version Mismatch Replies.
- Renamed the AUTH command to NOOP.
- 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

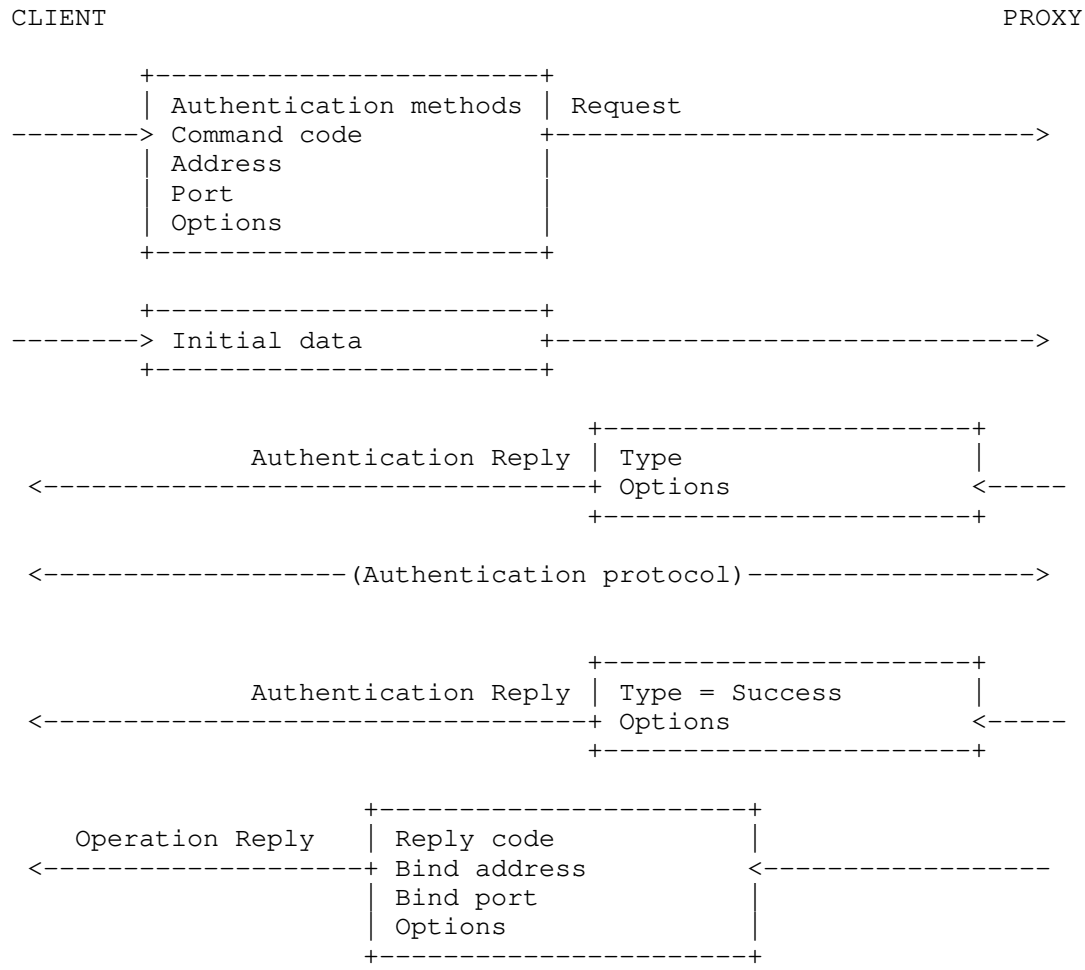

Figure 1: The SOCKS version 6 protocol message exchange

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 conection 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.

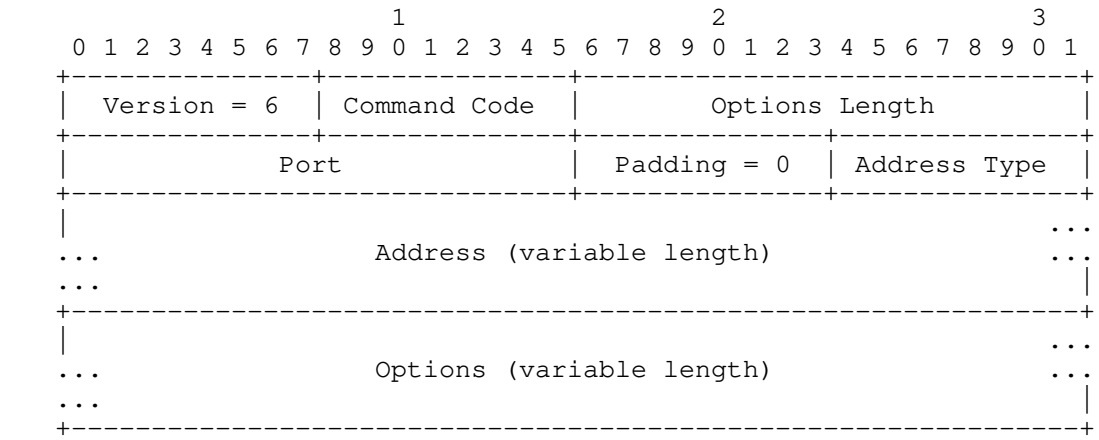

Figure 2: SOCKS 6 Request

- Version: 6
- 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.

- Address Type:
    * 0x01: IPv4
    * 0x03: Domain Name
    * 0x04: IPv6
- 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
- Port: the port in network byte order.
- Padding: set to 0
- Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.
- Options: see Section 8.

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

- 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.
- CONNECT: The fields signify the address and port to which the client wishes to connect.
- 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:

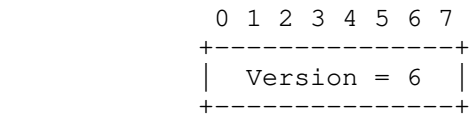

Figure 3: SOCKS 6 Version Mismatch Reply

- 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:

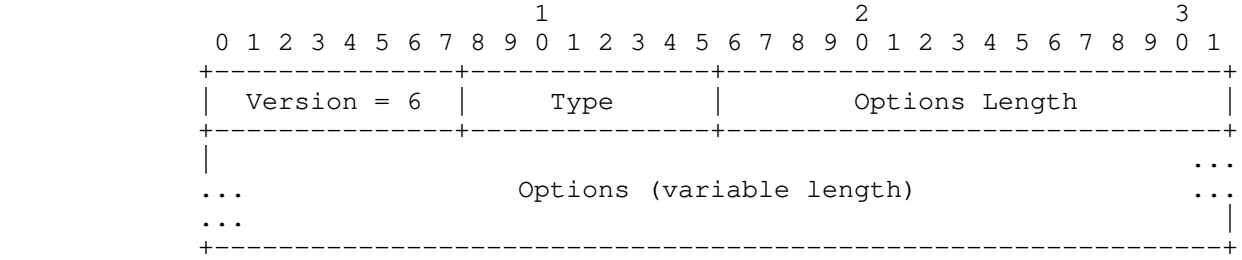

Figure 4: SOCKS 6 Authentication Reply

- Version: 6
- Type:
  - 0x00: authentication successful.
  - 0x01: authentication failed.
- Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.
- 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:

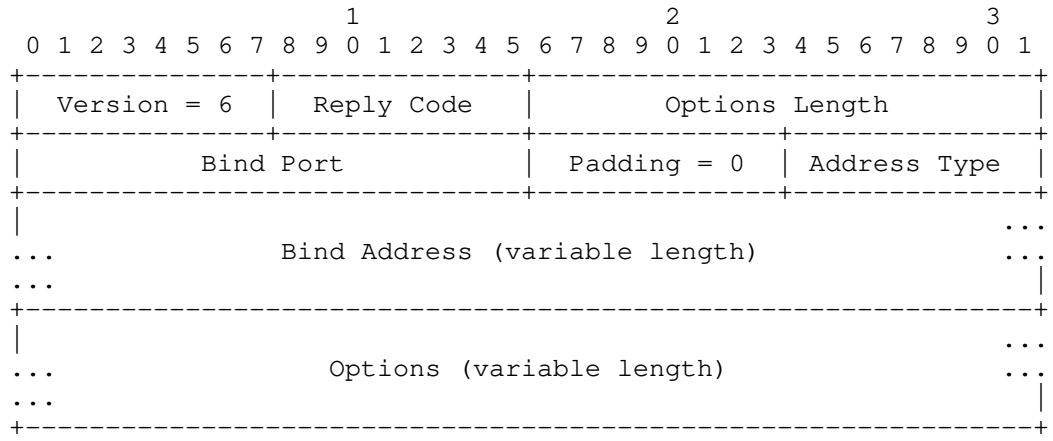

Figure 5: SOCKS 6 Operation Reply

- Version: 6
- Reply Code:
  - 0x00: Succes
  - 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

- Bind Port: the proxy bound port in network byte order.
- Padding: set to 0
- Address Type:
    * 0x01: IPv4
    * 0x03: Domain Name
    * 0x04: IPv6
- 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
- Options Length: the total size of the SOCKS options that appear in the Options field. MUST NOT exceed 16KB.
- 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:

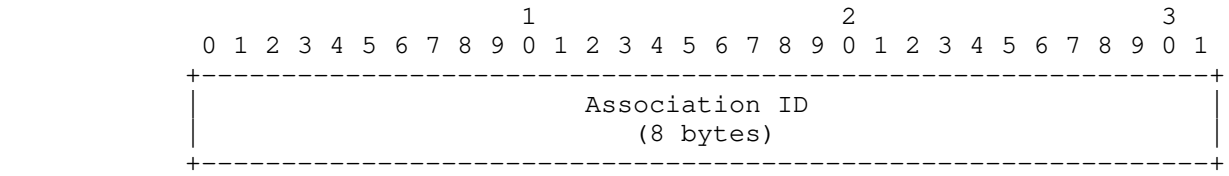

Figure 6: UDP Association Initialization

- 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:

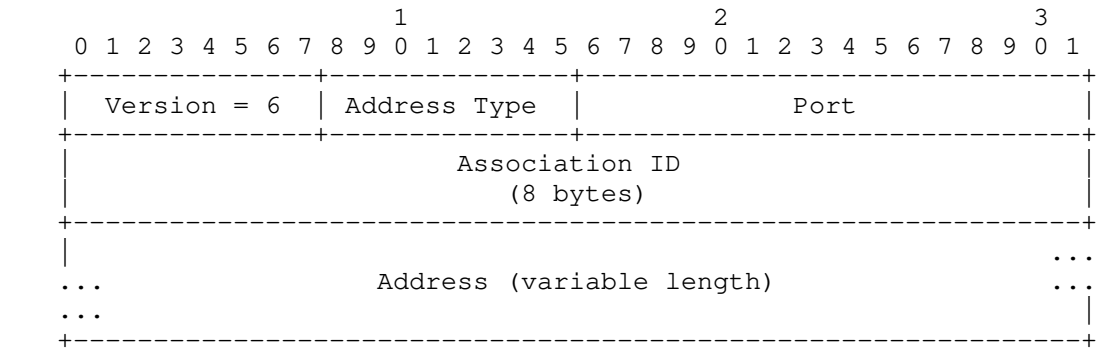

Figure 7: SOCKS 6 Datagram Header

- Version: 0x06
- Association ID: the identifier of the UDP association
- Address Type:
  - 0x01: IPv4
  - 0x03: Domain Name
  - 0x04: IPv6
- 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
- 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:

- the 5-tuple of the UDP conversation, if the datagram was received over plain UDP, or
- 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:

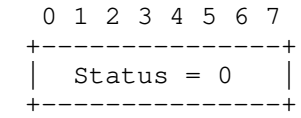

Figure 8: UDP Association Confirmation

- 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 addessses is permitted for UDP traffic only.

8. SOCKS Options

SOCKS options have the following format:

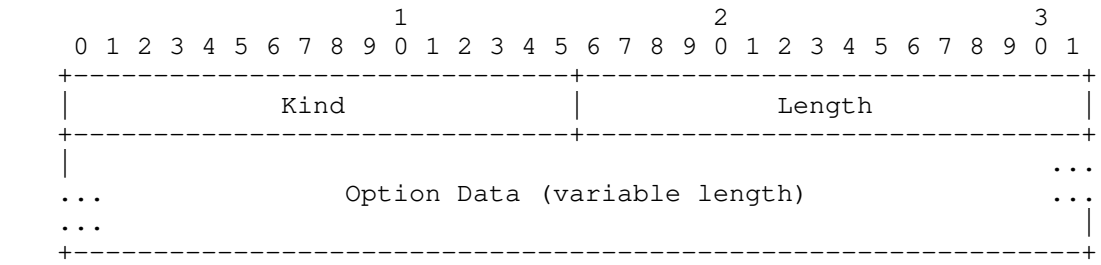

Figure 9: SOCKS 6 Option

- Kind: Allocated by IANA. (See Section 13.)
- Length: The total length of the option. MUST be a multiple of 4.
- 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 protcols 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.

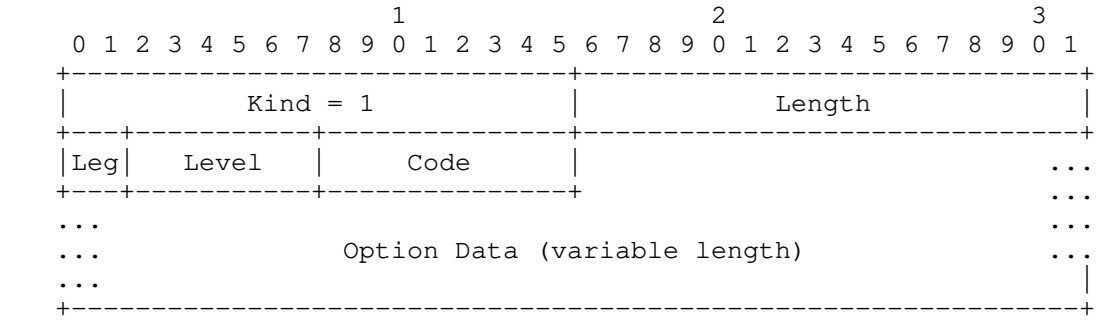

Figure 10: Stack Option

- Leg:
  - 1: Client-Proxy Leg
  - 2: Proxy-Remote Leg
  - 3: Both Legs
- Level:
  - 1: IP: options that apply to either IPv4 or IPv6
  - 2: IPv4
  - 3: IPv6
  - 4: TCP
  - 5: UDP
- Code: Option code
- Option Data: Option-specific data

8.1.1. IP TOS options

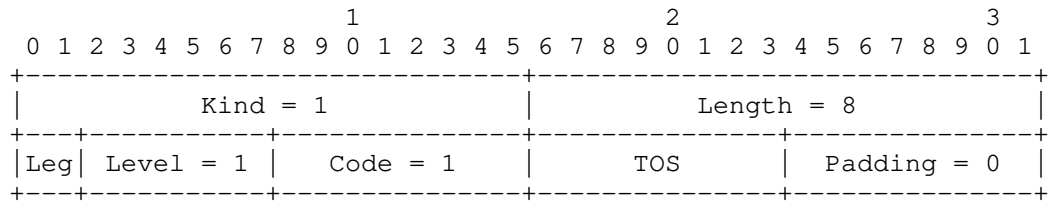

Figure 11: IP TOS Option

- 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. TFO options

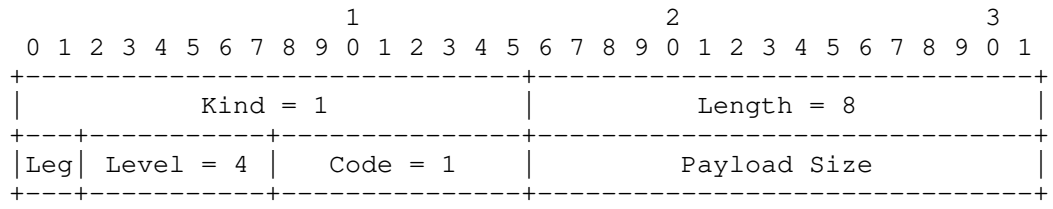

Figure 12: TFO Option

- Leg: 0x02 (Proxy-Remote leg).
- 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.

8.1.3. 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.

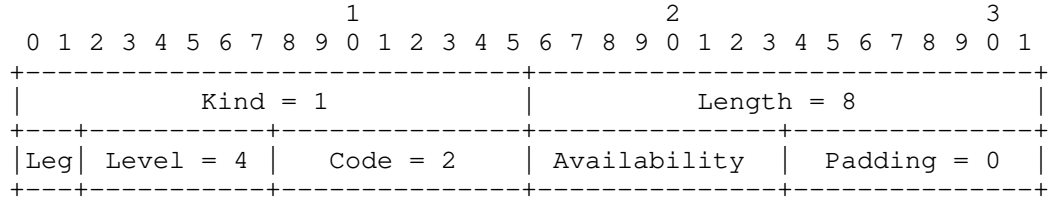

Figure 13: Multipath Option

- Leg: 0x02 (Proxy-Remote leg)
- 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.

8.1.4. Listen Backlog options

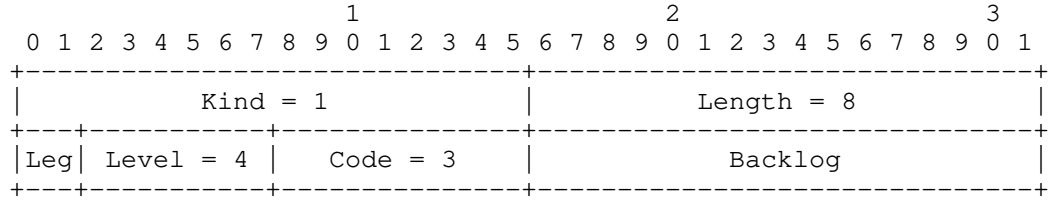

Figure 14: Listen Backlog Option

- Leg: 0x02 (Proxy-Remote leg)

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.

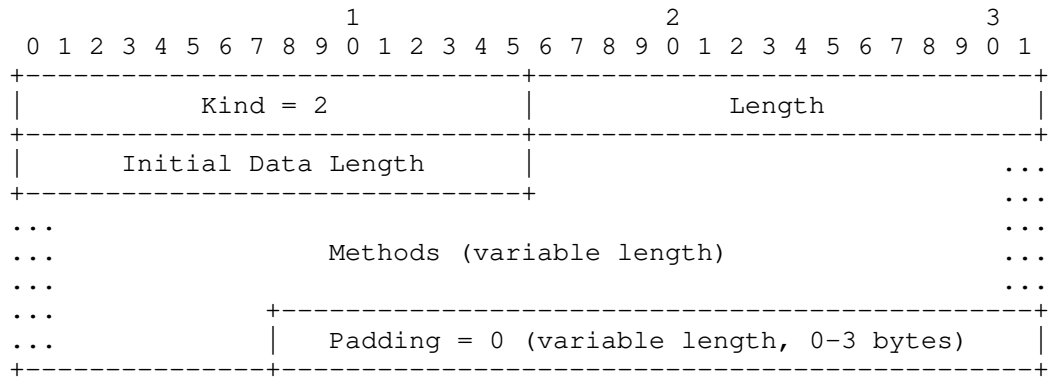

Figure 15: Authentication Method Advertisement Option

- 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.

- Methods: One byte per advertised method. Method numbers are assigned by IANA.

- 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:

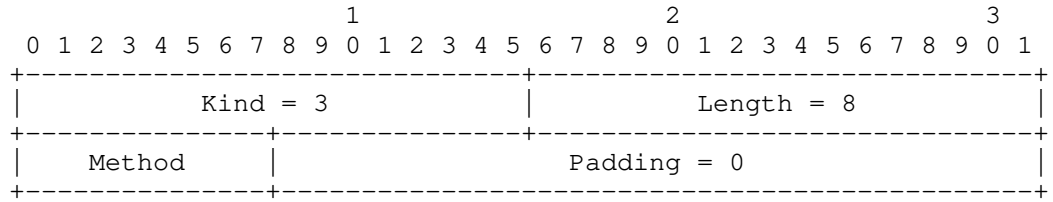

Figure 16: Authentication Method Selection Option

- 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:

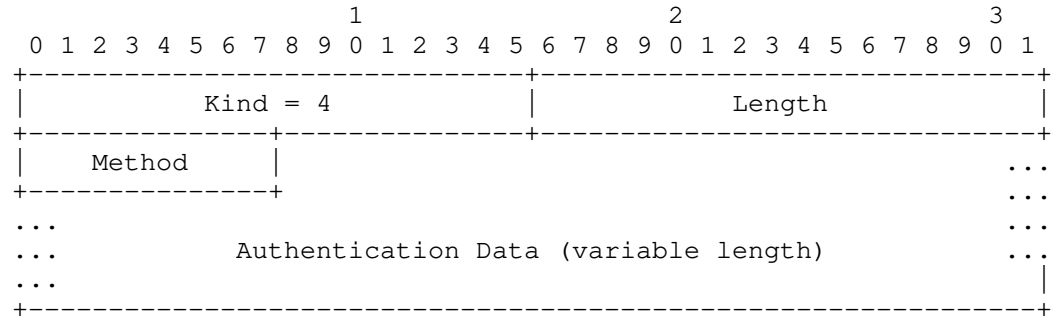

Figure 17: Authentication Data Option

- Method: The number of the authentication method. These numbers are assigned by IANA.

- 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:

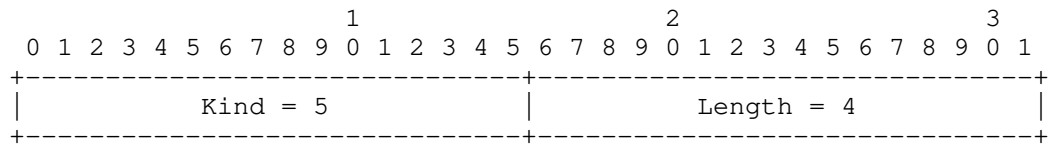

Figure 18: Session Request Option

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

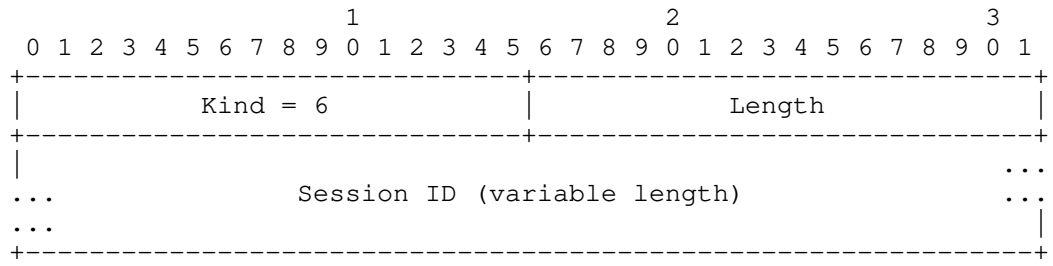

Figure 19: Session ID Option

- 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:

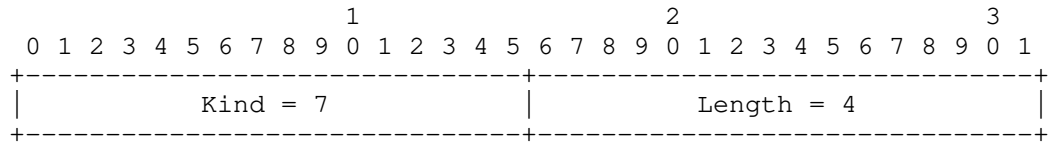

Figure 20: Session Untrusted Option

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 19).

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

For valid Session IDs:

- 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.

- 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:

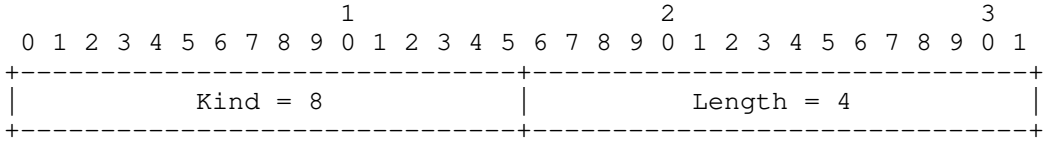

Figure 21: Session OK Option

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:

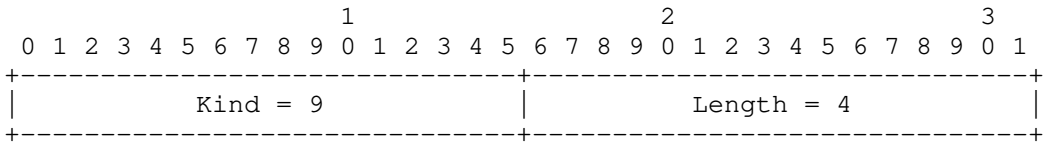

Figure 22: Session Invalid Option

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:

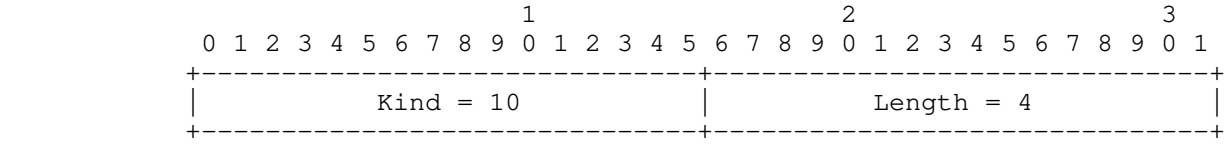

Figure 23: Session Teardown Option

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:

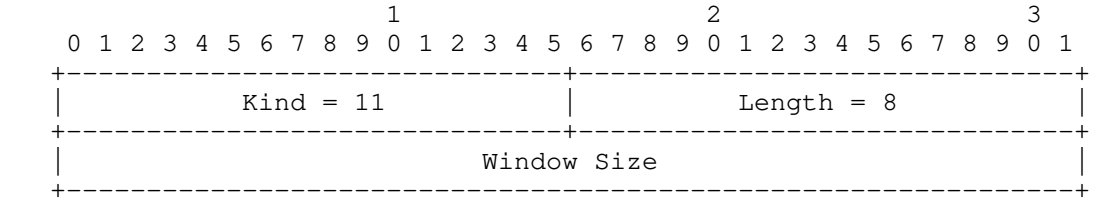

Figure 24: Token Request

- 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:

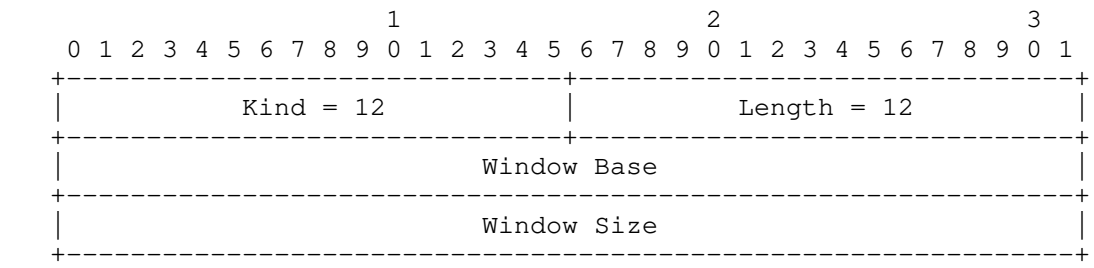

Figure 25: Idempotence Window

- Window Base: The first token in the window.

- 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:

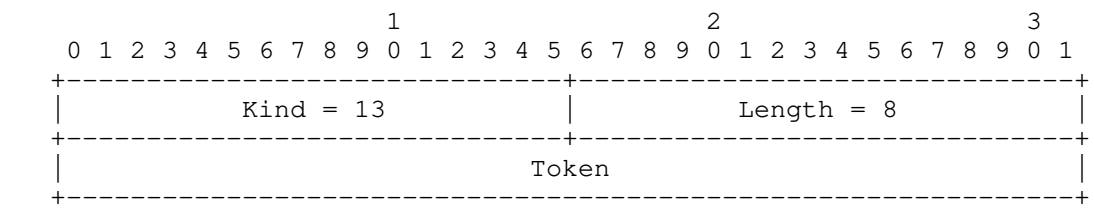

Figure 26: Idempotence Expenditure

- Kind: 13 (Idempotence Expenditure option)
- Length: 8
- 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:

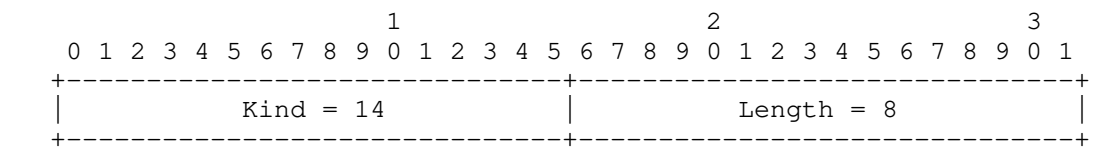

Figure 27: Idempotence Accepted

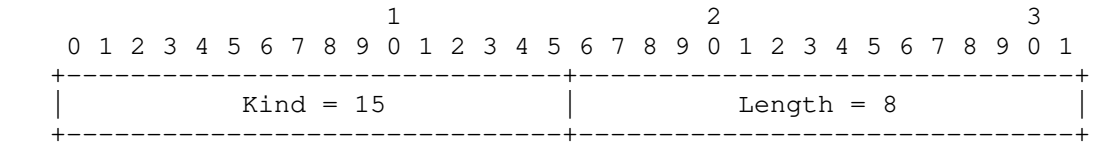

Figure 28: Idempotence Rejected

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.

8.6. Address Resolution Options

Clients wishing to resolve an address from the proxy's vantage point can do so by sending a Resolution Request option:

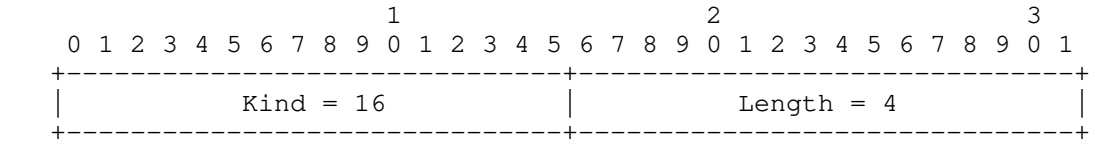

Figure 29: Resolution Request

The proxy's reply is included in the Operation Reply.

8.6.1. Forward resolution

If the address supplied by the client is a Domain Name, the proxy sends a Resolution option for each supported address type:

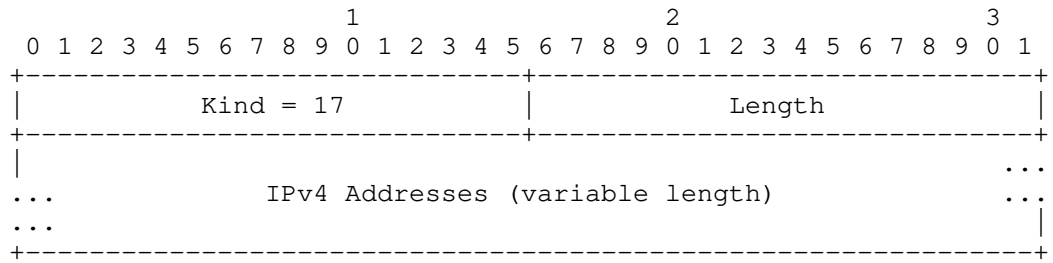

Figure 30: IPv4 Resolution

o IPv4 Addresses: The resolved IPv4 addresses.

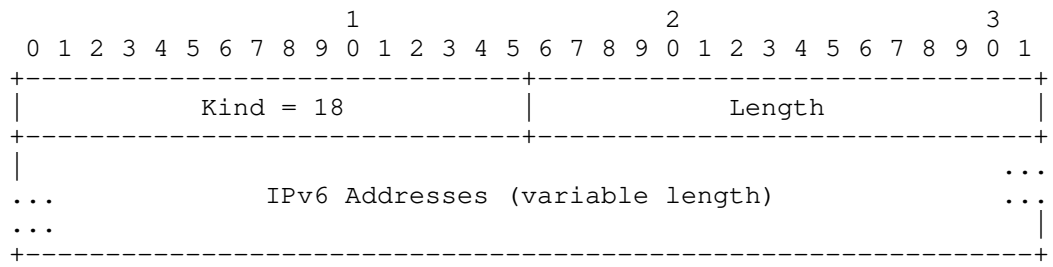

Figure 31: IPv6 Resolution

o IPv6 Addresses: The resolved IPv6 addresses.

If resolving a given address type is supported, but no addresses were found, the proxy MUST send an empty option, rather than none at all. If the proxy does not support resolving either one of the address types, it MUST omit sending the corrresponding option.

8.6.2. Reverse resolution

If the client supplies either an IPv4 or IPv6 address, the proxy performs a reverse lookup and replies with a list of domain names:

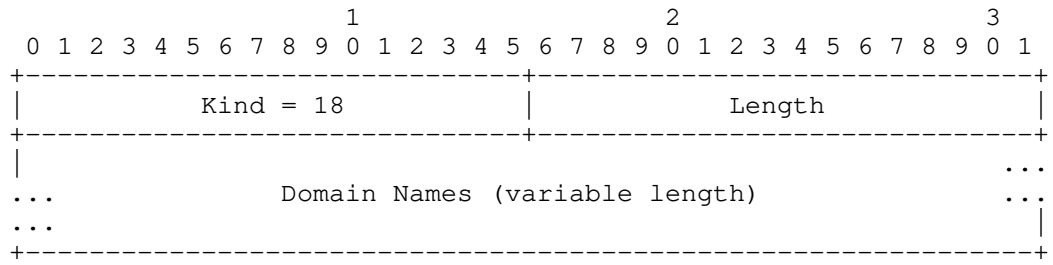

Figure 32: Domain Name Resolution

- Domain Names: The resolved domain names. Their format is the same as the one used in Requests and Operation Replies, with each string being individially padded. (See Section 4.)

If no domain names could be found, the proxy MUST send an empty option. If reverse resolution is unsupported, the proxy MUST NOT send any such option.

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.

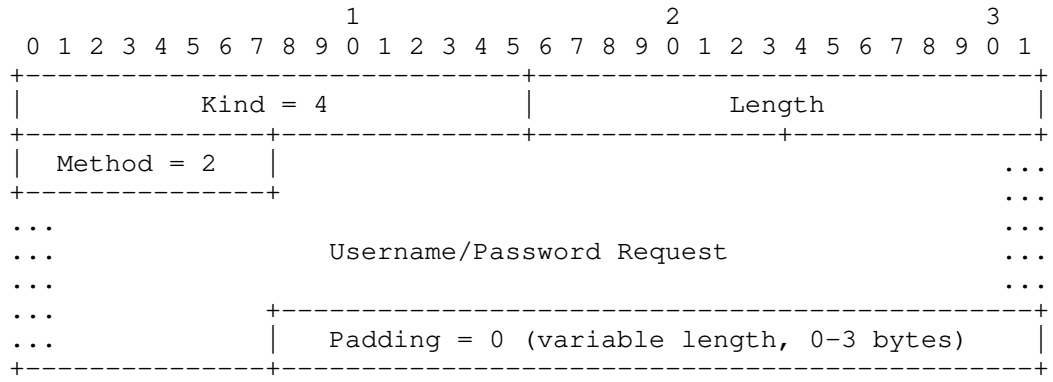

Figure 33: Password authentication via a SOCKS Option

- 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:

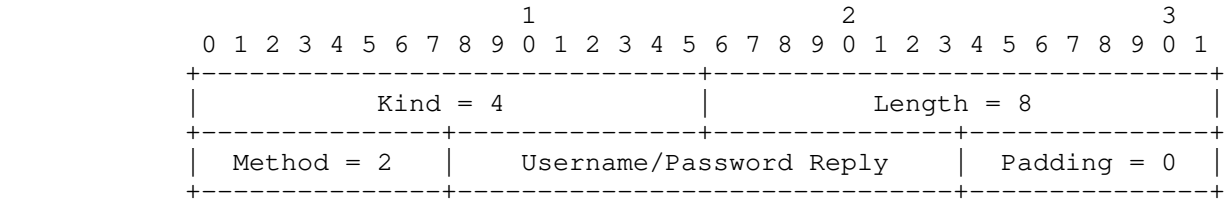

Figure 34: Reply to password authentication via a SOCKS Option

- 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:

- The protocol running on top of SOCKS tolerates the risks of TFO, or
- 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
- The client uses Idempotence Options, making replays very unlikely, or
- 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:

- 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.

- Application data MAY be sent immediately after receiving an Authentication Reply indicating success.

- When performing a method-specific authentication sequence, application data MAY be sent immediately after the last client message.

12. Security Considerations

12.1. 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:

- the request is not fully received after a certain timeout, or

- the number of options or their size exceeds an imposed hard cap.

12.2. 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.

12.3. 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.

13. IANA Considerations

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

- Unassigned: 0
- Stack: 1
- Authentication Method Advertisement: 2
- Authentication Method Selection: 3
- Authentication Data: 4
- Session Request: 5
- Session ID: 6
- Session Untrusted: 7
- Session OK: 8
- Session Invalid: 9
- Session Teardown: 10
- Idempotence Request: 11
- Idempotence Window: 12
- Idempotence Expenditure: 13
- Idempotence Accepted: 14
- Idempotence Rejected: 15
- Resolution Request: 16
- IPv4 Resolution: 17
- IPv6 Resolution: 18
- Domain Name Resolution: 19
- Vendor-specific: 64512-0xFFFF

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

14. Acknowledgements

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

15. References

15.1. 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>.

15.2. 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-06 (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

Network Working Group
Internet-Draft
Updates: 2863 (if approved)
Intended status: Standards Track
Expires: January 8, 2020

D. Thaler
Microsoft
D. Romascanu
Independent

July 07, 2019

Guidelines and Registration Procedures for Interface Types and Tunnel Types
draft-thaler-iftype-reg-04

Abstract

The registration and use of interface types ("ifType" values) predated the use of IANA Considerations sections and YANG modules, and so confusion has arisen about the interface type allocation process. Tunnel types were then added later, with the same requirements and allocation policy as interface types. This document provides updated guidelines for the definition of new interface types and tunnel types, for consideration by those who are defining, registering, or evaluating those definitions.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

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

This Internet-Draft will expire on January 8, 2020.

Copyright Notice

Table of Contents

1. Introduction

The IANA IfType-MIB was originally defined in [RFC1573] as a separate MIB module together with the Interfaces Group MIB (IF-MIB) module. The IF-MIB module has since been updated and is currently specified in [RFC2863], but this latest IF-MIB RFC no longer contains the IANA IfType-MIB. Instead, the IANA IfType-MIB is now maintained as a separate module. Similarly, [RFC7224] created an initial IANA Interface Type YANG Module, and the current version is maintained by IANA.

The current IANA IfType registry is at [ifType-registry], with the same values also appearing in [yang-if-type], and the IANAifType textual convention at [IANAifType-MIB].

Although the ifType registry was originally defined in a MIB module, the assignment and use of interface type values are not tied to MIB modules or any other management mechanism. Interface type values can be used as values of data model objects (MIB objects, YANG objects, etc.), as parts of a unique identifier of a data model for a given

interface type (e.g., in an OID), or simply as values exposed by local APIs or UI on a device.

The TUNNEL-MIB was then defined in [RFC2667] (now obsoleted by [RFC4087]) which created a tunnelType registry ([tunnelType-registry] and the IANAtunnelType textual convention at [IANAifType-MIB]) and defined the assignment policy for tunnelType values to always be identical to the policy for assigning ifType values.

2. Terminology

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. Problems

This document addresses the following issues:

1. As noted in Section 1, the former guidance was written with wording specific to MIB modules, and accordingly some confusion has resulted when using YANG modules. This document clarifies that ifTypes and tunnelTypes are independent from the type of, or even existence of, a data model.

2. The use of, and requirements around, sub-layers and sub-types are not well understood even though good examples of both exist. This is discussed in Section 4.

3. Since the ifType and tunnelType registries were originally defined, and are still retrievable, in the format of MIB modules (in addition to other formats), confusion arose when adding YANG modules as another format, as to whether each format is a separate registry. This is discussed in Section 5.

4. The registries are retrievable in the format of MIB and YANG modules, but there was no process guidance written to check that those formats were syntactically correct as updates were made, which led to the registry having syntax errors that broke tools. Section 6.1 adds a validation step to the documented assignment procedure.

5. Transmission values [transmission-registry] have often been allocated as part of ifType allocation, but no guidance exists about whether a requester must ask for it or not, and the request form has no such required field. As a result, IANA has asked the

Designated Expert to decide for each allocation, but no relevant guidance for the Designated Expert has been documented. This is remedied in Section 6.2.

6. Various documents and registries said to submit requests via email, but a web form exists for submitting requests, which caused some confusion around which is to be used. This is discussed in Section 7.

4. Interface Sub-Layers and Sub-Types

When multiple sub-layers exist below the network layer, each sub-layer can be represented by its own row in the ifTable with its own ifType, with the ifStackTable being used to identify the upward and downward multiplexing relationships between rows. Section 3.1.1 of [RFC2863] provides more discussion, and Section 3.1.2 of that RFC provides guidance for defining interface sub-layers. More recent experience shows that these guidelines are phrased in a way that is now too restrictive, since at the time [RFC2863] was written, MIB modules were the dominant data model.

This document clarifies that such guidance also applies to YANG modules.

Some ifTypes may define sub-types. For example, the tunnel(131) ifType defines sub-types, where each tunnelType can have its own MIB and/or YANG module with protocol-specific information, but there is enough in common that some information is exposed in a generic IP Tunnel MIB corresponding to the tunnel(131) ifType.

For requests that involve multiple sub-layers below the network layer, requests MUST include (or reference) a discussion of the multiplexing relationships between sub-layers, ideally with a diagram. Various well-written examples exist of such definitions, including [RFC3637] section 3.4.1, [RFC4087] section 3.1.1, and [RFC5066] section 3.1.1.

Definers of sub-layers and sub-types should consider which model is more appropriate for their needs. A sub-layer is generally used whenever either a dynamic relationship exists (i.e., which instances layer over which other instances can change over time) or a multiplexing relationship exists with another sub-layer. A sub-type can be used when neither of these are true, but where one interface type is conceptually a subclass of another interface type, as far as a management data model is concerned.

NEW CLARIFICATION/ELABORATION: In general, the intent of an interface type or sub-type is that its definition should be sufficient to

identify an interoperable protocol. In some cases, however, a protocol might be defined in a way that is not sufficient to provide interoperability with other compliant implementations of that protocol. In such a case, an ifType definition should discuss whether specific instantiations (or profiles) of behavior should use a sub-layer model (e.g., each vendor might layer the protocol over its own sub-layer that provides the missing details), or a sub-type model (i.e., each vendor might subclass the protocol without any layering relationship). If a sub-type model is more appropriate, then the data model for the protocol might include a sub-type identifier so that management software can discover objects specific to the subtype. In either case, such discussion is important to guide definers of a data model for the more specific information (i.e., a lower sub-layer or a subtype), as well as the Designated Expert that must review requests for any such ifTypes or sub-types.

4.1. Alternate Values

Another design decision is whether to reuse an existing ifType or tunnelType value, possibly using a sub-type or sub-layer model for refinements, or to use a different value for a new mechanism.

If there is already an entry that could easily satisfy the modeling and functional requirements for the requested entry, it should be reused so that applications and tools that use the existing value can be used without changes. If however, the modeling and functional requirements are significantly different enough such that having existing applications and tools use the existing value would be seen as a problem, a new value should be used.

For example, originally multiple ifType values were used for different flavors of Ethernet (ethernetCsmacd(6), iso88023Csmacd(7), fastEther(62), etc.), typically because they were registered by multiple vendors. [RFC3635] then deprecated all but ethernetCsmacd(6), since using different values was seen as problematic since all were functionally similar.

As another example, the Teredo tunnel protocol [RFC4380] encapsulates packets over UDP, and a udp(8) tunnelType value was defined in [RFC2667], with the description "The value UDP indicates that the payload packet is encapsulated within a normal UDP packet (e.g., RFC 1234)." However, the protocol model is quite different between [RFC1234] and Teredo. For example, [RFC1234] supports encapsulation of multicast/broadcast traffic whereas Teredo does not. As such, it would be more confusing to applications and tools to represent them using the same tunnel type, and so [RFC4087] defined a new value for Teredo.

In summary, definers of new interface or tunnel mechanisms should use a new ifType or tunnelType value rather than reusing an existing value when key aspects such as the header format or the link model (point-to-point, non-broadcast multi-access, broadcast capable multi-access, unidirectional broadcast, etc.) are significantly different from existing values, but reuse the same value when the differences can be expressed in terms of differing values of existing objects, other than ifType/tunnelType, in the standard YANG or MIB module.

5. Available Formats

Many registries are available in multiple formats. For example, XML, HTML, CSV, and Plain text are common formats in which many registries are available. This document clarifies that the [IANAifType-MIB], [yang-if-type], and [yang-tunnel-type] MIB and YANG modules are merely additional formats in which the ifType and tunnelType registries are available. The MIB and YANG modules are not separate registries, and the same values are always present in all formats of the same registry.

CURRENT: The confusion stems in part due to the fact that the IANA "Protocol Registries" [protocol-registries] lists the YANG and MIB module formats separately, as if they were separate registries. However, the entries for the yang-if-type and iana-tunnel-type YANG modules say "See ifType definitions registry." and "See tunnelType registry (mib-2.interface.ifTable.ifEntry.ifType.tunnelType)." respectively, although the entry for the IANAifType-MIB has no such note.

PROPOSED: It is proposed to clarify the relationship for the ifType and tunnelType registries as follows:

1. Add the following note to the entry for the IANAifType-MIB at [protocol-registries]: "This is one of the available formats of the ifType and tunnelType registries."

2. Change the note on the entry for the iana-if-type YANG module at [protocol-registries] to read: "This is one of the available formats of the ifType registry."

3. Change the note on the entry for the iana-tunnel-type YANG module at [protocol-registries] to read: "This is one of the available formats of the tunnelType registry."

4. Create a section for "Interface Parameters" at [protocol-registries], with entries for "Interface Types (ifType)" [ifType-registry], "Tunnel Types (tunnelType)"

[tunnelType-registry], and "Transmission Values" [transmission-registry].

5. Update the ifType definitions registry [ifType-registry] to list MIB [IANAifType-MIB] and YANG [yang-if-type] as Available Formats.

6. Update the tunnelType definitions registry [tunnelType-registry] to list MIB [IANAifType-MIB] and YANG [yang-tunnel-type] as Available Formats, and change the title to "tunnelType Definitions" for consistency with the [ifType-registry] title.

7. Replace the [yang-if-type] page with the YANG module content, rather than having a page that itself claims to have multiple Available Formats.

8. Replace the [yang-tunnel-type] page with the YANG module content, rather than having a page that itself claims to have multiple Available Formats.

6. Registration

The IANA policy (using terms defined in [RFC8126]) for registration is Expert Review, for both Interface Types and Tunnel Types. The role of the Designated Expert in the procedure is to raise possible concerns about wider implications of proposals for use and deployment of interface types. While it is recommended that the responsible Area Director and the IESG take into consideration the Designated Expert opinions, nothing in the procedure empowers the Designated Expert to override properly arrived-at IETF or working group consensus.

6.1. Procedures

Someone wishing to register a new ifType or tunnelType value MUST:

1. Check the IANA registry to see whether there is already an entry that could easily satisfy the modeling and functional requirements for the requested entry. If there is already such an entry, use it or update the existing specification. Text in an Internet-Draft, or part of some other permanently available, stable specification may be written to clarify the usage of an existing entry or entries for the desired purpose.

2. Check the IANA registry to see whether there is already some other entry with the desired name. If there is already an unrelated entry under the name, choose a different name.

3. Prepare a registration request using the template specified in Section 6.3. The registration request can be contained in an Internet-Draft, submitted alone, or as part of some other permanently available, stable, specification. The registration request can also be submitted in some other form (as part of another document or as a stand-alone document), but the registration request will be treated as an "IETF Contribution" under the guidelines of [RFC5378].

4. Submit the registration request (or pointer to document containing it) to IANA at iana@iana.org or (if requesting an ifType) via the web form at https://www.iana.org/form/iftype .

Upon receipt of a registration request, the following steps MUST be followed:

1. IANA checks the submission for completeness; if required information is missing or any citations are not correct, IANA will reject the registration request. A registrant can resubmit a corrected request if desired.

2. IANA requests Expert Review of the registration request against the corresponding guidelines from this document.

3. The Designated Expert will evaluate the request against the criteria.

4. Once the Designated Expert approves registration, IANA updates [ifType-registry], [IANAifType-MIB], and [yang-if-type] to show the registration for an Interface Type, or [tunnelType-registry], [IANAifType-MIB], and [yang-tunnel-type] to show the registration for a Tunnel Type. When adding values, IANA should verify that the updated MIB module and YANG module formats are syntactically correct before publishing the update. There are various existing tools or web sites that can be used to do this verification.

5. If instead the Designated Expert does not approve registration (e.g., for any of the reasons in [RFC8126] section 3), a registrant can resubmit a corrected request if desired, or the IESG can override the Designated Expert and approve it per the process in Section 5.3 of [RFC8126].

6.2. Media-specific OID-subtree assignments

The current IANAifType-MIB notes:

> The relationship between the assignment of ifType values and of OIDs to particular media-specific MIBs is solely the purview of

IANA and is subject to change without notice. Quite often, a media-specific MIB's OID-subtree assignment within MIB-II's 'transmission' subtree will be the same as its ifType value. However, in some circumstances this will not be the case, and implementors must not pre-assume any specific relationship between ifType values and transmission subtree OIDs.

The following change is to be made:

OLD: For every ifType registration, the corresponding transmission number value should be registered or marked "Reserved."

NEW: For future ifType assignments, an OID-subtree assignment MIB-II's 'transmission' subtree will be made with the same value.

Rationale: (1) This saves effort in the future since if a transmission number is later needed, no IANA request is needed that would then require another Expert Review. (2) The transmision numbering space is not scarce, so there seems little need to reserve the number for a different purpose than what the ifType is for. (3) The Designated Expert need not review whether a transmission number value should be registered when processing each ifType request, thus reducing the possibility of delaying assignment of ifType values. (4) There is no case on record where allocating the same value could have caused any problem.

6.3. Registration Template

6.3.1. ifType

The following template describes the fields that MUST be supplied in a registration request suitable for adding to the ifType registry:

Label for IANA ifType requested: As explained in Section 7.1.1 of [RFC2578], a label for a named-number enumeration must consist of one or more letters or digits, up to a maximum of 64 characters, and the initial character must be a lower-case letter. (However, labels longer than 32 characters are not recommended.) Note that hyphens are not allowed.

Name of IANA ifType requested: A short description (e.g., a protocol name), suitable to appear in a comment in the registry.

Description of the proposed use of the IANA ifType: Requesters MUST include answers, either directly or via a link to some document with the answers, to the following questions in the explanation of the proposed use of the IANA IfType:

- How would IP run over your ifType?

- Would there be another interface sublayer between your ifType and IP?

- Would your ifType be vendor-specific or proprietary? (If so, the label MUST start with a string that shows that. For example, if your company's name or acronym is xxx, then the ifType label would be something like xxxSomeIfTypeLabel.)

- (NEW) Would your ifType require or allow vendor-specific extensions? If so, would the vendor use their own ifType in a sub-layer below the requested ifType, or a sub-type of the ifType, or some other mechanism?

Reference, Internet-Draft, or Specification: A link to some document is required.

Additional information or comments: Optionally any additional comments for IANA or the Designated Expert.

6.3.2. tunnelType

CURRENT: No form exists for tunnelType, and it is not enforced that new requests have to use the ifType form.

PROPOSED: The following template describes the fields that MUST be supplied in a registration request suitable for adding to the tunnelType registry:

Label for IANA tunnelType requested: As explained in Section 7.1.1 of [RFC2578], a label for a named-number enumeration must consist of one or more letters or digits, up to a maximum of 64 characters, and the initial character must be a lower-case letter. (However, labels longer than 32 characters are not recommended.) Note that hyphens are not allowed.

Name of IANA tunnelType requested: A short description (e.g., a protocol name), suitable to appear in a comment in the registry.

Description of the proposed use of the IANA tunnelType: Requesters MUST include answers, either directly or via a link to some document with the answers, to the following questions in the explanation of the proposed use of the IANA tunnelType:

- How would IP run over your tunnelType?

- Would there be another interface sublayer between your tunnelType and IP?

- Would your tunnelType be vendor-specific or proprietary? (If so, the label MUST start with a string that shows that. For example, if your company's name or acronym is xxx, then the tunnelType label would be something like xxxSomeIfTypeLabel.)

- Would your tunnelType require or allow vendor-specific extensions? If so, would the vendor use their own tunnelType in a sub-layer below the requested tunnelType, or some sort of sub-type of the tunnelType, or some other mechanism?

Reference, Internet-Draft, or Specification: A link to some document is required.

Additional information or comments: Optionally any additional comments for IANA or the Designated Expert.

7. Submitting Requests

[IANAifType-MIB] currently says: "Requests for new values should be made to IANA via email (iana&iana.org)." However, a web form exists (https://www.iana.org/form/iftype), which is an apparent contradiction, but submissions either way are accepted.

IANA is requested to update the MIB module to instead say: "Interface types must not be directly added to the IANAifType-MIB MIB module. They must instead be added to the "ifType definitions" registry at [ifType-registry]."

(Note that [yang-if-type] was previously updated with this language.)

8. IANA Considerations

This entire document is about IANA considerations.

9. Security Considerations

Since this document does not introduce any technology or protocol, there are no security issues to be considered for this document itself.

10. References

10.1. 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>.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, DOI 10.17487/RFC2578, April 1999, <https://www.rfc-editor.org/info/rfc2578>.

[RFC2863] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, DOI 10.17487/RFC2863, June 2000, <https://www.rfc-editor.org/info/rfc2863>.

[RFC5378] Bradner, S., Ed. and J. Contreras, Ed., "Rights Contributors Provide to the IETF Trust", BCP 78, RFC 5378, DOI 10.17487/RFC5378, November 2008, <https://www.rfc-editor.org/info/rfc5378>.

[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>.

10.2. Informative References

[IANAifType-MIB]
IANA, "IANAifType-MIB", February 2019, <http://www.iana.org/assignments/ianaiftype-mib>.

[ifType-registry]
IANA, "ifType definitions", June 2019, <https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-5>.

[protocol-registries]
IANA, "Protocol Registries", June 2019, <https://www.iana.org/protocols>.

[RFC1234] Provan, D., "Tunneling IPX traffic through IP networks", RFC 1234, DOI 10.17487/RFC1234, June 1991, <https://www.rfc-editor.org/info/rfc1234>.

[RFC1573] McCloghrie, K. and F. Kastenholz, "Evolution of the Interfaces Group of MIB-II", RFC 1573, DOI 10.17487/RFC1573, January 1994, <https://www.rfc-editor.org/info/rfc1573>.

[RFC2667] Thaler, D., "IP Tunnel MIB", RFC 2667, DOI 10.17487/RFC2667, August 1999, <https://www.rfc-editor.org/info/rfc2667>.

[RFC3635] Flick, J., "Definitions of Managed Objects for the Ethernet-like Interface Types", RFC 3635, DOI 10.17487/RFC3635, September 2003, <https://www.rfc-editor.org/info/rfc3635>.

[RFC3637] Heard, C., Ed., "Definitions of Managed Objects for the Ethernet WAN Interface Sublayer", RFC 3637, DOI 10.17487/RFC3637, September 2003, <https://www.rfc-editor.org/info/rfc3637>.

[RFC4087] Thaler, D., "IP Tunnel MIB", RFC 4087, DOI 10.17487/RFC4087, June 2005, <https://www.rfc-editor.org/info/rfc4087>.

[RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", RFC 4380, DOI 10.17487/RFC4380, February 2006, <https://www.rfc-editor.org/info/rfc4380>.

[RFC5066] Beili, E., "Ethernet in the First Mile Copper (EFMCu) Interfaces MIB", RFC 5066, DOI 10.17487/RFC5066, November 2007, <https://www.rfc-editor.org/info/rfc5066>.

[RFC7224] Bjorklund, M., "IANA Interface Type YANG Module", RFC 7224, DOI 10.17487/RFC7224, May 2014, <https://www.rfc-editor.org/info/rfc7224>.

[transmission-registry] IANA, "transmission definitions", June 2019, <https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-7>.

[tunnelType-registry]
IANA, "Internet-standard MIB - mib-2.interface.ifTable.ifEntry.ifType.tunnelType", June 2019, <https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-6>.

[yang-if-type]
IANA, "iana-if-type YANG Module", February 2019, <http://www.iana.org/assignments/iana-if-type>.

[yang-tunnel-type]
IANA, "iana-tunnel-type YANG Module", June 2019, <https://www.iana.org/assignments/iana-tunnel-type>.

Authors' Addresses

Dave Thaler
Microsoft

EMail: dthaler@microsoft.com

Dan Romascanu
Independent

EMail: dromasca@gmail.com