Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 20 November 2026

T. Pauly
Apple, Inc.
D. Damjanovic
Microsoft
Y. Rosomakho
Zscaler

19 May 2026

Communicating Proxy Configurations in Provisioning Domains
draft-ietf-intarea-proxy-config-14

Abstract

This document defines a mechanism for accessing provisioning domain information associated with a proxy, such as other proxy URIs that support different protocols and information about which destinations are accessible using a proxy.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/tfpauly/privacy-proxy.

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 20 November 2026.

Copyright Notice

Table of Contents

1. Introduction

HTTP proxies that use the CONNECT method defined in Section 9.3.6 of [HTTP] (often referred to as "forward" proxies) allow clients to open connections to hosts via a proxy. These typically allow for TCP stream proxying, but can also support UDP proxying [CONNECT-UDP] and IP packet proxying [CONNECT-IP]. The locations of these proxies are not just defined as hostnames and ports, but can use URI templates [URITEMPLATE].

In order to make use of multiple related proxies, clients need a way to understand which proxies are associated with one another, and which protocols can be used to communicate with the proxies.

Clients can also benefit from learning about additional information associated with the proxy to optimize their proxy usage, such knowing that a proxy is configured to only allow access to a limited set of destinations.

These improvements to client behavior can be achieved through the use of Provisioning Domains. Provisioning Domains (PvDs) are defined in [PVD] as consistent sets of network configuration information, which can include proxy configuration details (Section 2 of [PVD]). Section 4.3 of [PVDDATA] defines a JSON [JSON] format for describing Provisioning Domain Additional Information, which is an extensible dictionary of properties of the Provisioning Domain.

This document defines several mechanisms to use PvDs to help clients understand how to use proxies:

1. A way to fetch PvD Additional Information associated with a known proxy URI (Section 2)

2. A way to list one or more proxy URIs in a PvD, allowing clients to learn about other proxy options given a known proxy (Section 3).

3. A way to define the set of destinations that are accessible through the proxy (Section 4).

Using this mechanism a client can learn that a legacy insecure HTTP proxy that the client is configured with is also accessible using HTTPS. In this way, clients can upgrade to a more secure connection to the proxy.

1.1. Background

Non-standard mechanisms for proxy configuration and discovery have been used historically, some of which are described in the informational [RFC3040]: Proxy Auto Configuration (PAC) files Section 6.2 of [RFC3040] are JavaScript scripts that take URLs as input and provide an output of a proxy configuration to use. Web Proxy Auto-Discovery Protocol (WPAD) Section 6.4 of [RFC3040] allows networks to advertise proxies to use by advertising a PAC file. This solution uses the DHCPv4 option 252, reserved for private use according to Section 2.1 of [IANA-DHCP]. These common (but non-standard) mechanisms only support defining proxies by hostname and port, and do not support configuring a full URI template [URITEMPLATE].

The mechanisms defined in this document are intended to offer a standard alternative that works for URI-based proxies and avoids dependencies on executing JavaScript scripts, which are prone to implementation inconsistencies and security vulnerabilities.

1.2. Requirements Keywords

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.

1.3. Note to the RFC Editor

RFC EDITOR: Please remove this section before publication.

Various identifier words are used in this draft using the code markdown and are easily noted in the HTML rendering of this draft. The authors kindly request that the RFC editor makes these instances noticeable via appropriate markings in the TXT and PDF renderings of this draft. The term include, but may not be limited to the following: proxies protocol proxy mandatory alpn identifier

2. Fetching PvD Additional Information for proxies

This document defines a way to fetch PvD Additional Information associated with a proxy. This PvD describes the properties of the network accessible through the proxy.

Clients fetch PvD Additional Information associated with a proxy by issuing an HTTP GET request for a PvD URI using the "application/pvd+json" media type as defined in Section 4.1 of [PVDDATA]. The fetch MUST use the "https" scheme.

[PVDDATA] defines the well-known PvD URI, that uses a path of "/.well-known/pvd" and is served on the standard port for HTTP over TLS (HTTPS), port 443. When a client is provisioned with the hostname of a proxy for which it wants to look up PvD Additional Information, the client SHALL use the well-known PvD URI using the host authority of the proxy. A client can also be directly configured with a HTTPS URI on which to fetch the PvD Information, in which case the fetch SHALL be made to that configured URI.

A client MAY cache the information it obtained from PvD Additional Information, but it MUST discard cached information if:

* The current time is beyond the "expires" value defined in Section 4.3 of [PVDDATA]

* A new Sequence Number for that PvD is received in a Router Advertisement (RA)

To avoid synchronized queries toward the server hosting the PvD Additional Information when an object expires, clients MUST apply a randomized backoff as specified in Section 4.1 of [PVDDATA].

For example, a client would issue the following request for the PvD associated with "https://proxy.example.org/masque{?target_host,target_port}":

```
:method = GET
:scheme = https
:authority = proxy.example.org
:path = /.well-known/pvd
accept = application/pvd+json
```

A client would send the same request as above for the PvD associated with an HTTP CONNECT proxy on "proxy.example.org:8080". Note that the client will not make the GET request for the PvD to port 8080, but to port 443.

Note that all proxies that are co-located on the same host share the same PvD Additional Information. Proxy deployments that need separate PvD configuration properties MUST use different hosts.

PvD Additional Information is required to contain the "identifier", "expires", and "prefixes" keys. For proxy PvDs as defined in this document, the "identifier" MUST match the hostname of the HTTP proxy. The "prefixes" array MUST be empty for cases when the PvD identifier is not provided by a Router Advertisement as defined in [PVDDATA].

2.1. Discovery via HTTPS/SVCB Records

To allow clients to determine whether PvD Additional Information is available for a particular named host (which allows fetching proxy information, as well as any other information in the PvD), this document defines a new SvcParamKey in HTTPS and SVCB DNS records defined in [SVCB-DNS].

Presence of this SvcParamKey, named pvd, indicates that the host supports PvD discovery via the well-known PvD URI defined in Section 4.1 of [PVDDATA]. The presence of this key in an HTTPS or SVCB record signals that PvD Additional Information can be fetched using the "https" scheme from the host on port 443 using the well-known path. The value of the pvd SvcParamKey MUST be empty.

A client receiving a DNS record like the following:

```
proxy.example.org. 3600 IN HTTPS 1 . alpn="h3,h2" pvd
```

can interpret the presence of the pvd key as an indication that it MAY perform a PvD fetch from "https://proxy.example.org/.well-known/pvd" using HTTP GET method.

This key is useful for detecting proxy configurations when looking up a DNS record for a known proxy name, but is a generic hint that PvD Additional Information is available. Future extensions to PvD Additional Information can also take advantage of this discovery mechanism.

This hint is advisory; clients MAY still attempt to fetch PvD Additional Information even if pvd SvcParamKey is not present.

The pvd SvcParamKey is registered with IANA as described in Section 7.5.

3. Enumerating proxies within a PvD

This document defines a new PvD Additional Information key, proxies, that is an array of dictionaries, where each dictionary in the array defines a single proxy that is available as part of the PvD (see Section 7.1). Each proxy is defined by a proxy protocol and a proxy location (i.e., a hostname and port or a URI template [URITEMPLATE]), along with other optional keys.

When a PvD that contains the proxies key is fetched from a known proxy using the method described in Section 2, the proxies array describes proxies that can be used in addition to the known proxy. The proxies may potentially supporting other protocols.

Such cases are useful for informing clients of related proxies as a discovery method, with the assumption that the client already is aware of one proxy. Many historical methods of configuring a proxy only allow configuring a single hostname and port for the proxy. A client can attempt to fetch the PvD information from the well-known URI to learn the list of complete URIs that support non-default protocols, such as [CONNECT-UDP] and [CONNECT-IP].

3.1. Proxy dictionary keys

This document defines two required keys for the sub-dictionaries in the proxies array: protocol and proxy. There are also optional keys, including mandatory, alpn, and identifier. Other optional keys (keys defined in future extensions or proprietary key defined in Section 3.2) can be added to the dictionary to further define or restrict the use of a proxy. The keys are registered with IANA as described in Section 7.2, with the initial content provided below.

JSON Key	Optional/ Required	Description	Type	Example
protocol	required	The protocol used to communicate with the proxy	String	"connect-udp"
proxy	required	String containing the URI template or host and port of the proxy,	String	"https://example.org:4443/masque/{?target_host,target_port}"

		depending on the format defined by the protocol		
mandatory	optional	An array of optional keys that client must understand and process to use this proxy	Array of Strings	["example_key"]
alpn	optional	An array of Application-Layer Protocol Negotiation protocol identifiers	Array of Strings	["h3","h2"]
identifier	optional	A string used to refer to the proxy, which can be referenced by other dictionaries, such as entries in proxy-match	String	"udp-proxy"

Table 1: Initial Proxy Information PvD Keys Registry Contents

The values for the protocol key are defined in the proxy protocol registry (Section 7.3), with the initial contents provided below. For consistency, any new proxy types that use HTTP Upgrade Tokens (and use the :protocol pseudo-header) MUST define the protocol value to match the Upgrade Token / :protocol value. Extensions to proxy types that use the same HTTP Upgrade Tokens ought to be covered by the same protocol value; if there are properties specific to an extension, the extensions can either define new optional keys or rely on negotiation within the protocol to discover support.

Proxy Protocol	Proxy Location Format	Reference	Notes
socks5	host:port	[SOCKSv5]	
http-connect	host:port	Section 9.3.6 of [HTTP]	Standard CONNECT method, using unencrypted HTTP to the proxy
https-connect	host:port	Section 9.3.6 of [HTTP]	Standard CONNECT method, using TLS-protected HTTP to the proxy
connect-udp	URI template	[CONNECT-UDP]	
connect-ip	URI template	[CONNECT-IP]	
connect-tcp	URI template	[CONNECT-TCP]	

Table 2: Initial PvD Proxy Protocol Registry Contents

The value of proxy depends on the Proxy Location Format defined by proxy protocol. The types defined here either use a host as defined in Section 3.2.2 of [URI] and port, or a full URI template.

The value of the mandatory key is an array of keys that the client must understand and process to be able to use the proxy. A client that does not understand a key from the array or cannot fully process the value of a key from the array MUST ignore the entire proxy dictionary.

The mandatory array can contain keys that are either:

* registered in an IANA registry, defined in Section 7.2 and marked as optional,

* or proprietary, as defined in Section 3.2

The mandatory array MUST NOT include any entries that are not present in the sub-dictionary.

If the alpn key is present, it provides a hint for the Application-Layer Protocol Negotiation (ALPN) [ALPN] protocol identifiers associated with this server. For HTTP proxies, this can indicate if the proxy supports HTTP/3, HTTP/2, etc.

The value of the identifier key is a string that can be used to refer to a particular proxy from other dictionaries, specifically those defined in Section 4. The string value is an arbitrary non-empty JSON string using UTF-8 encoding as discussed in Section 8.1 of [JSON]. Characters that need to be escaped in JSON strings per Section 7 of [JSON] are NOT RECOMMENDED as they can lead to difficulties in string comparisons as discussed in Section 8.3 of [JSON]. Identifier values MAY be duplicated across different proxy dictionaries in the proxies array. References to a particular identifier apply to the set of proxies sharing that identifier. Proxies without the identifier key are expected to accept any traffic since their destinations cannot be contained in proxy-match array defined in Section 4. Proxies with identifier keys are expected to accept traffic based on matching rules in the proxy-match array and MUST NOT be used if they are not included in the proxy-match array.

3.2. Proprietary keys in proxy configurations

Implementations MAY include proprietary or vendor-specific keys in the sub-dictionaries of the proxies array to convey additional proxy configuration information not defined in this specification.

A proprietary key MUST contain at least one underscore character ("_") as a delimiter in the string, with characters both before and after the underscore. The right-most underscore serves as a separator between a vendor-specific namespace and the key name; i.e., the string to the right of the right-most underscore is the key name and the string to the left of the right-most underscore specifies the vendor-specific namespace. For example, "example_tech_authmode" could be a proprietary key indicating an authentication mode defined by a vendor named "Example Tech".

When combined with mandatory array, this mechanism allows implementations to extend proxy metadata while maintaining interoperability and ensuring safe fallback behavior for clients that do not support a given extension.

3.3. Example

Given a known HTTP CONNECT proxy FQDN, "proxy.example.org", a client could request PvD Additional Information with the following request:

```
   :method = GET
   :scheme = https
   :authority = proxy.example.org
   :path = /.well-known/pvd
   accept = application/pvd+json
```

   If the proxy has a PvD definition for this FQDN, it would return the following response to indicate a PvD that has two related proxy URIs.

```
:status = 200
content-type = application/pvd+json
content-length = 322

{
  "identifier": "proxy.example.org.",
  "expires": "2026-06-23T06:00:00Z",
  "prefixes": [],
  "proxies": [
    {
      "protocol": "http-connect",
      "proxy": "proxy.example.org:80"
    },
    {
      "protocol": "connect-udp",
      "proxy": "https://proxy.example.org/masque{?target_host,target_port}"
    }
  ]
}
```

   From this response, the client would learn the URI template of the proxy that supports UDP using [CONNECT-UDP], at "https://proxy.example.org/masque{?target_host,target_port}".

4. Destination accessibility information for proxies

   Destination accessibility information is used when only a subset of destinations is reachable through a proxy. Destination restrictions are often used in VPN tunnel configurations such as split DNS in IKEv2 [IKEV2SPLIT], and in other proxy configuration mechanisms like PAC files (see Section 1.1).

   PvD Additional Information can be used to indicate that a set of proxies only allows access to a limited set of destinations.

   To support determining which traffic is supported by different proxies, this document defines a new PvD Additional Information key proxy-match. This key has a value that is an array of dictionaries, where each subdictionary describes a rule for matching traffic to one

or more proxies, or excluding the traffic from all proxies described in the PvD. These subdictionaries are referred to as "destination rules", since they define rules about which destinations can be accessed for a particular proxy or set of proxies.

4.1. Destination Rule Keys

This document defines four keys for destination rules. Any destination rule MUST contain the proxies key. Values corresponding to the proxies key may be either an empty array, which implies that no proxy defined in this PvD can process matching traffic, or an array of strings with at least one proxy identifier string. A destination rule MAY contain one or more additional keys that describe destination properties. If no destination property keys are present, the rule matches all destinations, subject to proxy protocol and proxy applicability checks described in Section 4.2. Each destination property key’s value MUST be an array with at least one entry.

Extensions or proprietary deployments can define new keys to describe destination properties. Any destination rules that include keys not known to the client, or values that cannot be parsed, MUST be ignored in their entirety.

Destination rule keys are registered with IANA as defined in Section 7.4, with the initial content provided below.

JSON Key	Optional	Description	Type	Example
proxies	No	An array of strings that match identifier values from the top-level proxies array	Array of Strings	["tcp-proxy", "udp-proxy"]
domains	Yes	An array of FQDNs and wildcard DNS domains	Array of Strings	["www.example.com", "*.internal.example.com"]
subnets	Yes	An array of IPv4 and IPv6 addresses and subnets	Array of Strings	["2001:db8::1", "192.0.2.0/24"]
ports	Yes	An array of TCP and UDP port ranges	Array of Strings	["80", "443", "1024-65535"]

Table 3: Initial PvD Proxy Destination Rule Registry Contents

The domains array includes specific FQDNs and zones that are either accessible using specific proxy (for rules with non-empty proxies array) or non-accessible through any proxies (for rules with empty proxies array). Wildcards are allowed only as prefixes (*.). A wildcard prefix is used to indicate matching entire domains or subdomains instead of specific hostnames. Note that this can be used to match multiple levels of subdomains. For example, "*.example.com" matches "internal.example.com" as well as "www.public.example.com". Entries that include the wildcard prefix also match an FQDN that only contains the string after the prefix, with no subdomain. So, an entry "*.example.com" in the domains array of a proxy-match rule would match the FQDN "example.com". This is done to prevent commonly needing to include both "*.example.com" and "example.com" in the domains array of a proxy-match rule. Matches are performed against absolute domain names, independent of the client's configured DNS search suffixes. Clients MUST NOT apply local DNS suffix search rules when interpreting domains entries. A string MAY have a trailing dot ("."); it does not affect the matching logic.

The subnets array includes IPv4 and IPv6 address literals, as well as IPv4 address subnets represented using CIDR notation [CIDR] and IPv6 address prefixes Section 2.3 of [IPv6-ADDR]. Subnet-based destination information can apply to cases where applications are communicating directly with an IP address (without having resolved a DNS name) as well as cases where an application resolved a DNS name to a set of IP addresses. Note that if destination rules include an empty proxies array (indicating that no proxy is applicable for this subnet), an application can only reliably follow this destination rule if it resolves DNS names prior to proxying.

The ports array includes specific ports (used for matching TCP and/or UDP ports), as well as ranges of ports written with a low port value and a high port value, with a - in between. For example, "1024-2048" matches all ports from 1024 to 2048, including port 1024 and 2048. If ports key is not present, all ports are assumed to match. The array may contain individual port numbers (such as "80") or inclusive ranges of ports.

4.2. Using Destination Rules

The destination rules can be used to determine which traffic can be sent through proxies, and which specific set of proxies to use for any particular connection. By evaluating the rules in order, a consistent behavior for usage can be achieved.

Rules in the proxy-match array are provided in order of priority, such that a client can evaluate the rules from the first in the array to the last in the array, and attempt using the matching proxy or

proxies from the earliest matching rule first. If earliest matching rule has empty array of proxies, a client MUST NOT send matching traffic to any proxy defined in this PvD.

In order to match a destination rule in the proxy-match array, all properties MUST apply. For example, if a destination rule includes a domains array and a ports array, traffic that matches the rule needs to match at least one of the entries in the domains array and one of the entries in the ports array. In addition, a destination rule is considered a match only if at least one of the associated proxy identifiers is supported by the client(client understand all mandatory keys in the protocol description) and supports the protocol required by the connection attempt (for example, connect-udp for UDP traffic). If no listed proxy identifier is applicable, the rule MUST be treated as not matching, and the client continues evaluation of subsequent rules.

A matched rule will then either point to one or more proxy identifier values, which correspond to proxies defined in the array from Section 3, or instructs the client to not send the matching traffic to any proxy. If a matching rule contains more than one identifier, the client MUST treat the array as an ordered list, where the first identifier is the most preferred. Multiple proxy dictionaries can contain the same identifier value. In this case, the client can choose any of the proxies; however, the client ought to prefer using the same proxy for the consecutive requests to the same proxy identifier to increase connection reuse.

Entries listed in a proxy-match object MUST NOT expand the set of destinations that a client is willing to send to a particular proxy. The array can only narrow the set of destinations that the client is willing to send through the proxy. For example, if the client has a local policy to only send requests for "*.example.com" to a proxy "proxy.example.com", and domains array of a match object contains "internal.example.com" and "other.company.com", the client would end up only proxying "internal.example.com" through the proxy.

4.3. Proprietary Keys in Destination Rules

Implementations MAY include proprietary or vendor-specific keys in destination rules to define custom matching logic not specified in this document.

Similar to proprietary keys in proxy dictionaries (Section 3.2), a proprietary key in destination rule MUST contain at least one underscore character ("_"), which separates a vendor-specific namespace from the key name. For example, "acme_processid" could be a key used to apply rules only to traffic of a specific process identifier as defined by a vendor named "acme".

Clients that encounter a proprietary key they do not recognize MUST ignore the entire destination rule in which the key appears. This ensures that unknown or unsupported matching logic does not inadvertently influence proxy selection or bypass security controls.

4.4. Examples

In the following example, two proxies are defined with a common identifier ("default_proxy"), with a single destination rule for "*.internal.example.org".

```
{
  "identifier": "proxy.example.org.",
  "expires": "2026-06-23T06:00:00Z",
  "prefixes": [],
  "proxies": [
    {
      "protocol": "http-connect",
      "proxy": "proxy.example.org:80",
      "identifier": "default_proxy"
    },
    {
      "protocol": "http-connect",
      "proxy": "proxy2.example.org:80",
      "identifier": "default_proxy"
    }
  ],
  "proxy-match": [
    {
      "domains": [ "*.internal.example.org" ],
      "proxies": [ "default_proxy" ]
    }
  ]
}
```

The client could then choose to use either proxy associated with the "default_proxy" identifier for accessing TCP hosts that fall within the "*.internal.example.org" zone. This would include the hostnames "internal.example.org", "foo.internal.example.org", "www.bar.internal.example.org" and all other hosts within "internal.example.org". The client will use the same proxy for the

following requests to hosts falling into the "*.internal.example.org" zone to increase connection reuse and make use of the connection resumption. The client will not use the proxies defined in this configuration to hosts outside of the "*.internal.example.org" zone.

In the next example, two proxies are defined with a distinct identifier, and there are three destination rules:

```
{
  "identifier": "proxy.example.org.",
  "expires": "2026-06-23T06:00:00Z",
  "prefixes": [],
  "proxies": [
    {
      "protocol": "http-connect",
      "proxy": "proxy.example.org:80",
      "identifier": "default_proxy"
    },
    {
      "protocol": "http-connect",
      "proxy": "special-proxy.example.org:80",
      "identifier": "special_proxy"
    }
  ],
  "proxy-match": [
    {
      "domains": [ "*.special.example.org" ],
      "ports": [ "80", "443", "49152-65535" ],
      "proxies": [ "special_proxy" ]
    },
    {
      "domains": [ "no-proxy.internal.example.org" ],
      "proxies": [ ]
    },
    {
      "domains": [ "*.internal.example.org" ],
      "proxies": [ "default_proxy" ]
    }
  ]
}
```

In this case, the client would use "special-proxy.example.org:80" for any TCP traffic that matches "*.special.example.org" destined to ports 80, 443 or any port between 49152 and 65535. The client would not use any of the defined proxies for access to "no-proxy.internal.example.org". And finally, the client would use "proxy.example.org:80" to access any other TCP traffic that matches "*.internal.example.org".

In the following example, three proxies are sharing a common identifier ("default-proxy"), but use separate protocols constraining the traffic that they can process.

```
{
  "identifier": "proxy.example.org.",
  "expires": "2026-06-23T06:00:00Z",
  "prefixes": [],
  "proxies": [
    {
      "protocol": "http-connect",
      "proxy": "proxy.example.org:80",
      "identifier": "default_proxy"
    },
    {
      "protocol": "connect-udp",
      "proxy": "https://proxy.example.org/masque/udp/{target_host},{target_port}",
      "identifier": "default_proxy"
    },
    {
      "protocol": "connect-ip",
      "proxy": "https://proxy.example.org/masque/ip{?target,ipproto}",
      "identifier": "default_proxy"
    }
  ],
  "proxy-match": [
    {
      "domains": [ "*.internal.example.org" ],
      "proxies": [ "default_proxy" ]
    }
  ]
}
```

The client would use proxies in the following way:

* Traffic not destined to hosts within the "*.internal.example.org" zone is not sent to any proxy defined in this configuration

* TCP traffic destined to hosts within the "*.internal.example.org" zone is sent either to the proxy with "http-connect" protocol or to the proxy with "connect-ip" protocol

* UDP traffic destined to hosts within the "*.internal.example.org" zone is sent either to the proxy with "connect-udp" protocol or to the proxy with "connect-ip" protocol

* Traffic other than TCP and UDP destined to hosts within the "*.internal.example.org" zone is sent to the proxy with "connect-ip" protocol

The following example provides a configuration of proxies to be used by default with a set with exceptions to bypass:

```
{
  "identifier": "proxy.example.org.",
  "expires": "2026-06-23T06:00:00Z",
  "prefixes": [],
  "proxies": [
    {
      "protocol": "http-connect",
      "proxy": "proxy.example.org:80",
      "identifier": "default_proxy"
    },
    {
      "protocol": "http-connect",
      "proxy": "backup.example.org:80",
      "identifier": "secondary_proxy"
    }
  ],
  "proxy-match": [
    {
      "domains": [ "*.intranet.example.org" ],
      "proxies": [ ]
    },
    {
      "subnets": [ "192.0.2.0/24", "2001:db8::/32" ],
      "proxies": [ ]
    },
    {
      "proxies": [ "default_proxy", "secondary_proxy" ]
    }
  ]
}
```

In this case, the client will not forward TCP traffic that is destined to hosts matching "*.intranet.example.org", 192.0.2.0/24 or 2001:db8::/32, through the proxies. Due to the order in "proxies" array in the last rule of "proxy-match", the client would prefer "proxy.example.org:80" over "backup.example.org:80"

The following example provides a configuration of proxies that enable setting one proxy for "example.org" and a different proxy for all of its subdomains, i.e. "*.example.org":

```
{
  "identifier": "proxy.example.org.",
  "expires": "2026-06-23T06:00:00Z",
  "prefixes": [],
  "proxies": [
    {
      "protocol": "http-connect",
      "proxy": "proxy1.example.org:80",
      "identifier": "proxy1"
    },
    {
      "protocol": "http-connect",
      "proxy": "proxy2.example.org:80",
      "identifier": "proxy2"
    }
  ],
  "proxy-match": [
    {
      "domains": [ "example.org" ],
      "proxies": [ "proxy1" ]
    },
    {
      "domains": [ "*.example.org" ],
      "proxies": [ "proxy2" ]
    }
  ]
}
```

In this case, the client will forward TCP traffic that is destined to host "example.org" to "proxy1.example.org:80" and all traffic to the subdomains of "example.org", i.e. "*.example.org" will be forwarded to "proxy2.example.org:80".

5. Discovering proxies from network PvDs

[PVDDATA] defines how PvD Additional Information is discovered based on network advertisements using Router Advertisements [RFC4861]. This means that a network defining its configuration via PvD information can include the proxies key (Section 3). However, clients MUST NOT automatically use these proxy configurations, unless the device has been explicitly provisioned to trust this configuration from the network for specific proxy hosts; for example, a corporate-managed device could use this mechanism on an authenticated corporate network to learn which of an allowed set of proxy URIs are available at this particular location.

Future specifications can define ways to dynamically trust proxy configurations delivered by a network, but such mechanisms are out of scope for this document.

6. Security Considerations

This document extends the PvD Additional Information defined in [PVDDATA]; as such, all security considerations from [PVDDATA] apply here.

The mechanisms in this document allow clients using a proxy to "upgrade" a configuration for a cleartext HTTP/1.1 or SOCKS proxy into a configuration that uses TLS to communication to the proxy. This upgrade can add protection to the proxied traffic so it is less observable by entities along the network path; however it does not prevent the proxy itself from observing the traffic being proxied.

Configuration advertised via PvD Additional Information, such as DNS zones or associated proxies, can only be safely used when fetched over a secure TLS-protected connection, and the client has validated that the hostname of the proxy, the identifier of the PvD, and the validated hostname identity on the certificate all match.

The lists of proxies and destination rules provided by the PvD Additional Information might exceed the memory constraints or processing capabilities of clients, particularly for constrained devices. A client that is not able to process all of the content of either the proxies list or destination rules due to resource limitations MUST ignore the proxy configuration entirely. Clients MUST implement limits for the maximum number of proxy configurations and destination rules that they are able to process; the specific limits will vary based on device capabilities.

When using information in destination rules (Section 4), clients MUST only allow the PvD configuration to narrow the scope of traffic that they will send through a proxy. Clients that are configured by policy to only send a particular set of traffic through a particular proxy can learn about rules that will cause them to send more narrowly-scoped traffic, but MUST NOT send traffic that would go beyond what is allowed by local policy.

As described in Section 5, proxy configuration discovered based on RAs from a network MUST NOT be automatically used by clients to start using proxies when they would otherwise not proxy traffic.

7. IANA Considerations

7.1. New PvD Additional Information key

This document registers two new keys in the "Additional Information PvD Keys" registry [IANA_PVD].

7.1.1. proxies Key

JSON Key: proxies

Description: Array of proxy dictionaries associated with this PvD

Type: Array of dictionaries

Example:

```
[
 {
  "protocol": "connect-udp",
  "proxy": "https://proxy.example.org/masque{?target_host,target_port}"
 }
]
```

7.1.2. proxy-match Key

JSON Key: proxy-match

Description: Array of proxy match rules, as dictionaries, associated with entries in the proxies array.

Type: Array of dictionaries

Example:

```
[
 {
  "domains": [ "*.internal.example.org" ],
  "proxies": [ "default_proxy" ]
 }
]
```

7.2. New PvD Proxy Information Registry

IANA is requested to create a new registry "Proxy Information PvD Keys", within the "Provisioning Domains (PvDs)" registry page. This new registry reserves JSON keys for use in sub-dictionaries under the proxies key. The initial contents of this registry are given in Table 1.

New assignments in the "Proxy Information PvD Keys" registry will be administered by IANA through Expert Review [RFC8126]. Experts are requested to ensure that defined keys do not overlap in names or semantics, do not contain an underscore character ("_") in the names (since underscores are reserved for vendor-specific keys), and have clear format definitions. The reference and notes fields may be empty.

7.3. New PvD Proxy Protocol Registry

IANA is requested to create a new registry "Proxy Protocol PvD Values", within the "Provisioning Domains (PvDs)" registry page. This new registry reserves JSON values for the protocol key in proxies sub-dictionaries. The initial contents of this registry are given in Table 2.

New assignments in the "Proxy Protocol PvD Values" registry will be administered by IANA through Expert Review [RFC8126]. Experts are requested to ensure that defined keys do not overlap in names. The reference and notes fields may be empty.

7.4. New PvD Proxy Destination Rule Registry

IANA is requested to create a new registry "Proxy Destination Rule PvD Keys", within the "Provisioning Domains (PvDs)" registry page. This new registry reserves JSON keys for use in sub-dictionaries under the proxy-match key. The initial contents of this registry are given in Table 3.

New assignments in the "Proxy Destination Rule PvD Keys" registry will be administered by IANA through Expert Review [RFC8126]. Experts are requested to ensure that defined keys do not overlap in names or semantics, and do not contain an underscore character ("_") in the names (since underscores are reserved for vendor-specific keys).

7.5. New DNS SVCB Service Parameter Key (SvcParamKey)

IANA is requested to add a new entry to the "DNS SVCB Service Parameter Keys (SvcParamKeys)" registry [IANA_SVCB]:

* Number: TBD

* Name: pvd

* Meaning: PvD configuration is available at the well-known path

* Change Controller: IETF

* Reference: this document, Section 2.1

8. References

8.1. Normative References

[ALPN] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/rfc/rfc7301>.

[CIDR] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <https://www.rfc-editor.org/rfc/rfc4632>.

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[IPv6-ADDR] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/rfc/rfc4291>.

[JSON] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[PVDDATA] Pfister, P., Vyncke, ., Pauly, T., Schinazi, D., and W. Shao, "Discovering Provisioning Domain Names and Data", RFC 8801, DOI 10.17487/RFC8801, July 2020, <https://www.rfc-editor.org/rfc/rfc8801>.

[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/rfc/rfc2119>.

[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/rfc/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/rfc/rfc8174>.

[SVCB-DNS] Schwartz, B., Bishop, M., and E. Nygren, "Service Binding and Parameter Specification via the DNS (SVCB and HTTPS Resource Records)", RFC 9460, DOI 10.17487/RFC9460, November 2023, <https://www.rfc-editor.org/rfc/rfc9460>.

[URI] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[URITEMPLATE] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/rfc/rfc6570>.

8.2. Informative References

[CONNECT-IP] Pauly, T., Ed., Schinazi, D., Chernyakhovsky, A., Khlewind, M., and M. Westerlund, "Proxying IP in HTTP", RFC 9484, DOI 10.17487/RFC9484, October 2023, <https://www.rfc-editor.org/rfc/rfc9484>.

[CONNECT-TCP] Schwartz, B. M., "Template-Driven HTTP CONNECT Proxying for TCP", Work in Progress, Internet-Draft, draft-ietf-httpbis-connect-tcp-11, 20 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-connect-tcp-11>.

[CONNECT-UDP] Schinazi, D., "Proxying UDP in HTTP", RFC 9298, DOI 10.17487/RFC9298, August 2022, <https://www.rfc-editor.org/rfc/rfc9298>.

[IANA-DHCP]
Droms, R., "Procedures and IANA Guidelines for Definition of New DHCP Options and Message Types", BCP 43, RFC 2939, DOI 10.17487/RFC2939, September 2000, <https://www.rfc-editor.org/rfc/rfc2939>.

[IANA_PVD] "Additional Information PvD Keys Registry", n.d., <https://www.iana.org/assignments/pvds/pvds.xhtml#additional-information-pvd-keys>.

[IANA_SVCB]
"SvcParamKeys Registry", n.d., <https://www.iana.org/assignments/dns-svcb/dns-svcb.xhtml#dns-svcparamkeys>.

[IKEV2SPLIT]
Pauly, T. and P. Wouters, "Split DNS Configuration for the Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 8598, DOI 10.17487/RFC8598, May 2019, <https://www.rfc-editor.org/rfc/rfc8598>.

[PVD] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <https://www.rfc-editor.org/rfc/rfc7556>.

[RFC3040] Cooper, I., Melve, I., and G. Tomlinson, "Internet Web Replication and Caching Taxonomy", RFC 3040, DOI 10.17487/RFC3040, January 2001, <https://www.rfc-editor.org/rfc/rfc3040>.

[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/rfc/rfc4861>.

[SOCKSv5] 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/rfc/rfc1928>.

Acknowledgments

The authors would like to thank Phil Lisiecki, David Schinazi, Dale Worley, Tomas Dragoun, Keith Holleman, James Taft, Tommy Jensen, and Ryan Globus for their detailed reviews, feedback, and suggestions during the development of this document.

Authors' Addresses

Tommy Pauly
Apple, Inc.
Email: tpauly@apple.com

Dragana Damjanovic
Microsoft
Email: ddamjanovic@microsoft.com

Yaroslav Rosomakho
Zscaler
Email: yrosomakho@zscaler.com