MASQUE
Internet-Draft
Intended status: Standards Track
Expires: 14 October 2026

D. Schinazi
Google LLC
Y. Rosomakho
Zscaler

12 April 2026

# DNS and PREF64 Configuration for Proxying IP in HTTP
# draft-ietf-masque-connect-ip-dns-06

## Abstract

Proxying IP in HTTP allows building a VPN through HTTP load balancers. However, at the time of writing, that mechanism lacks a way to communicate network configuration details such as DNS information or IPv6 NAT64 prefixes (PREF64). In contrast, most existing VPN protocols provide mechanisms to exchange such configuration information.

This document defines extensions that convey DNS and PREF64 configuration using HTTP capsules. This mechanism supports encrypted DNS transports and can be used to inform peers about network translation prefixes for IPv6/IPv4 synthesis.

## About This Document

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

The latest revision of this draft can be found at https://ietf-wg-masque.github.io/draft-ietf-masque-connect-ip-dns/draft-ietf-masque-connect-ip-dns.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-masque-connect-ip-dns/.

Discussion of this document takes place on the Multiplexed Application Substrate over QUIC Encryption Working Group mailing list (mailto:masque@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/masque/. Subscribe at https://www.ietf.org/mailman/listinfo/masque/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-masque/draft-ietf-masque-connect-ip-dns.

## Status of This Memo

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

This Internet-Draft will expire on 14 October 2026.

## Copyright Notice



## Table of Contents

## 1. Introduction

Proxying IP in HTTP ([CONNECT-IP]) allows building a VPN through HTTP load balancers. However, at the time of writing, that mechanism lacks a way to communicate network configuration details such as DNS information or IPv6 NAT64 prefixes (PREF64). In contrast, most existing VPN protocols provide mechanisms to exchange such configuration information (for example [IKEv2] supports discovery of DNS servers).

This document defines extensions that allow CONNECT-IP peers to convey DNS and PREF64 configuration information to its peer using HTTP capsules ([HTTP-DGRAM]). These extensions do not define any new way to transport DNS queries or responses, but instead enable the exchange of DNS resolver configuration and NAT64 prefix information inline with CONNECT-IP sessions.

Note that these extensions are meant for cases where CONNECT-IP operates as a Remote Access VPN (see Section 8.1 of [CONNECT-IP]) or a Site-to-Site VPN (see Section 8.2 of [CONNECT-IP]), but not for cases like IP Flow Forwarding (see Section 8.3 of [CONNECT-IP]).

For DNS configuration, this specification uses Service Bindings ([SVCB]) to exchange information about nameservers, such as which encrypted DNS transport is supported. This allows support for DNS over HTTPS ([DoH]), DNS over QUIC ([DoQ]), DNS over TLS ([DoT]), unencrypted DNS over UDP port 53 and TCP port 53 ([DNS]), as well as potential future DNS transports. PREF64 configuration is represented in a way similar to Router Advertisement option defined in Section 4 of [PREF64-RA].

Similar to how Proxying IP in HTTP exchanges IP address configuration information (Section 4.7 of [CONNECT-IP]), the mechanism defined in this document leverages capsules. Similarly, these mechanisms are bidirectional: either endpoint can send configuration information.

## 2. Conventions and Definitions

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.

This document uses terminology from [QUIC]. Where this document defines protocol types, the definition format uses the notation from Section 1.3 of [QUIC]. This specification uses the variable-length integer encoding from Section 16 of [QUIC]. Variable-length integer values do not need to be encoded in the minimum number of bytes necessary.

In this document, we use the term "nameserver" to refer to a DNS recursive resolver as defined in Section 6 of [DNS-TERMS], and the term "domain name" is used as defined in Section 2 of [DNS-TERMS].

## 3. DNS Configuration Mechanism

Either endpoint can send DNS configuration information in a DNS_ASSIGN capsule. The capsule format is defined below.

### 3.1. Domain Structure

```
Domain {
  Domain Length (i),
  Domain Name (..),
}
```

Figure 1: Domain Format

Each Domain contains the following fields:

Domain Length: Length of the following Domain field, encoded as a variable-length integer.

Domain Name: Fully Qualified Domain Name in DNS presentation format and using an Internationalized Domain Names for Applications (IDNA) A-label ([IDNA]).

### 3.2. Nameserver Structure

```
Nameserver {
  Service Priority (16),
  IPv4 Address Count (i),
  IPv4 Address (32) ...,
  IPv6 Address Count (i),
  IPv6 Address (128) ...,
  Authentication Domain Name (..),
  Service Parameters Length (i),
  Service Parameters (..),
}
```

Figure 2: Nameserver Format

Each Nameserver structure contains the following fields:

Service Priority: The priority of this attribute compared to other nameservers, as specified in Section 2.4.1 of [SVCB]. Since this specification relies on using Service Bindings in ServiceMode (Section 2.4.3 of [SVCB]), this field MUST NOT be set to 0.

IPv4 Address Count: The number of IPv4 Address fields following this field. Encoded as a variable-length integer.

IPv4 Address: Sequence of IPv4 Addresses that can be used to reach this nameserver. Encoded in network byte order.

IPv6 Address Count: The number of IPv6 Address fields following this field. Encoded as a variable-length integer.

IPv6 Address: Sequence of IPv6 Addresses that can be used to reach this nameserver. Encoded in network byte order.

Authentication Domain Name: A Domain structure (see Section 3.2) representing the domain name of the nameserver. This MAY be empty if the nameserver only supports unencrypted DNS (as traditionally sent over UDP port 53 and TCP port 53).

Service Parameters Length: Length of the following Service Parameters field, encoded as a variable-length integer.

Service Parameters: Set of service parameters that apply to this nameserver. Encoded using the wire format specified in Section 2.2 of [SVCB].

Service parameters allow exchanging additional information about the nameserver:

* The "port" service parameter is used to indicate which port the nameserver is reachable on. If no "port" service parameter is included, this indicates that default port numbers should be used.

* The "alpn" service parameter is used to indicate which encrypted DNS transports are supported by this nameserver. If the "no-default-alpn" service parameter is omitted, that indicates that the nameserver supports unencrypted DNS, as is traditionally sent over UDP port 53 and TCP port 53. In that case, the sum of IPv4 Address Count and IPv6 Address Count MUST be nonzero. If Authentication Domain Name is empty, the "alpn" and "no-default-alpn" service parameter MUST be omitted.

* The "dohpath" service parameter is used to convey a relative DNS over HTTPS URI Template, see Section 5 of [SVCB-DNS].

* The service parameters MUST NOT include "ipv4hint" or "ipv6hint" SvcParams, as they are superseded by the included IP addresses.

### 3.3. DNS Configuration Structure

```
DNS Configuration {
  Nameserver Count (i),
  Nameserver (..) ...,
  Internal Domain Count (i),
  Internal Domain (..) ...,
  Search Domain Count (i),
  Search Domain (..) ...,
}
```

Figure 3: DNS Configuration Format

Each DNS Configuration contains the following fields:

Nameserver Count:
: The number of Nameserver structures following this field. Encoded as a variable-length integer.

Nameserver:
: A series of Nameserver structures representing nameservers.

Internal Domain Count:
: The number of Domain structures following this field. Encoded as a variable-length integer.

Internal Domain:
: A series of Domain structures representing internal domain names.

Search Domain Count:
: The number of Domain structures following this field. Encoded as a variable-length integer.

Search Domain:
: A series of Domain structures representing search domains.

### 3.4. DNS_ASSIGN Capsule

The DNS_ASSIGN capsule (see Section 6 for the value of the capsule type) allows an endpoint to send DNS configuration to its peer.

```
DNS_ASSIGN Capsule {
  Type (i) = DNS_ASSIGN,
  Length (i),
  DNS Configuration (..) ...,
}
```

Figure 4: DNS_ASSIGN Capsule Format

DNS_ASSIGN capsule MAY include multiple DNS configurations if different DNS servers are responsible for separate internal domains.

If multiple DNS_ASSIGN capsules are sent in one direction, each DNS_ASSIGN capsule supersedes prior ones.

### 3.5. Handling

Note that internal domains include subdomains. In other words, if the DNS configuration contains a domain, that indicates that the corresponding domain and all of its subdomains can be resolved by the nameservers exchanged in the same DNS configuration. Sending an empty string as an internal domain indicates the DNS root; i.e., that the corresponding nameserver can resolve all domain names.

As with other IP Proxying capsules, the receiver can decide whether to use or ignore the configuration information. For example, in the consumer VPN scenario, clients will trust the IP proxy and apply received DNS configuration, whereas IP proxies will ignore any DNS configuration sent by the client.

If the IP proxy sends a DNS_ASSIGN capsule containing a DNS over HTTPS nameserver, then the client can validate whether the IP proxy is authoritative for the origin of the URI template. If this validation succeeds, the client SHOULD send its DNS queries to that nameserver directly as independent HTTPS requests. When possible, those requests SHOULD be coalesced over the same HTTPS connection.

### 3.6. Examples

#### 3.6.1. Full-Tunnel Consumer VPN

A full-tunnel consumer VPN hosted at masque.example.org could configure the client to use DNS over HTTPS to the IP proxy itself by sending the following configuration.

```
DNS Configuration = {
  Nameservers = [{
    Service Priority = 1,
    IPv4 Address = [],
    IPv6 Address = [],
    Authentication Domain Name = "masque.example.org",
    Service Parameters = {
      alpn=h2,h3
      dohpath=/dns-query{?dns}
    },
  }],
  Internal Domains = [""],
  Search Domains = [],
}
```

Figure 5: Full Tunnel Example

### 3.6.2. Split-Tunnel Enterprise VPN

An enterprise switching their preexisting IKEv2/IPsec split-tunnel VPN to connect-ip could use the following configuration.

```
DNS Configuration = {
  Nameservers = [{
    Service Priority = 1,
    IPv4 Address = [192.0.2.33],
    IPv6 Address = [2001:db8::1],
    Authentication Domain Name = "",
    Service Parameters = {},
  }],
  Internal Domains = ["internal.corp.example"],
  Search Domains = [
    "internal.corp.example",
    "corp.example",
  ],
}
```

Figure 6: Split Tunnel Example

## 4. PREF64 Configuration Mechanism

This document defines a new capsule type, PREF64, that allows a CONNECT-IP peer to communicate the IPv6 NAT64 prefixes to be used for IPv6/IPv4 address synthesis. This information enables an endpoint operating in an IPv6-only environment to construct IPv4-reachable addresses from IPv6 literals when a NAT64 translator is in use.

### 4.1. PREF64 Capsule

Each PREF64 capsule conveys zero or more NAT64 prefixes. If multiple capsules are sent in the same direction, the most recent one replaces any previously advertised prefixes. Empty PREF64 capsule is used to inform that NAT64 prefixes are not available.

The capsule has the following structure (see Section 6 for the value of the capsule type):

```
PREF64 Capsule {
  Type (i) = PREF64,
  Length (i),
  NAT64 Prefix (104) ...,
}
```

Figure 7: PREF64 Capsule Format

Individual NAT64 prefix has the following structure:

```
NAT64 Prefix {
  Prefix Length (8),
  Prefix (96),
}
```

Figure 8: NAT64 Prefix Format

NAT64 prefix fields are:

Prefix Length: The length of the NAT64 prefix in bits encoded as a single byte. Valid values are 32, 40, 48, 56, 64, and 96. These lengths correspond to the prefix sizes permitted for the IPv6-to-IPv4 address synthesis algorithm described in Section 2.2 of [IPv6-TRANSLATOR].

Prefix: The highest 96 bits of the IPv6 prefix, encoded in network byte order. Note that this field is always 96 bits long, regardless of the value in the Prefix Length field preceding it, see Section 2.2 of [IPv6-TRANSLATOR] for details.

### 4.2. Handling

Upon receiving a PREF64 capsule, a peer updates its local NAT64 configuration for the corresponding CONNECT-IP session. The newly received PREF64 capsule overrides any previously received PREF64 capsules in the same direction.

If an endpoint receives a capsule that does not meet one of the requirements listed in Section 4.1, or with a length that is not a multiple of 13 bytes, it MUST treat it as malformed. An empty PREF64 capsule invalidates any previously received NAT64 Prefixes.

### 4.3. Example

A CONNECT-IP peer would use the following capsule to signal a single NAT64 prefix of 64:ff9b::/96

```
PREF64 Capsule {
  Type (i) = PREF64,
  Length (i) = 13,
  NAT64 Prefix {
    Prefix Length (8) = 96
    Prefix (96) = 0x00 0x64 0xff 0x9b 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00,
  }
}
```

Figure 9: PREF64 Capsule Example

## 5. Security Considerations

Acting on received DNS_ASSIGN capsules can have significant impact on endpoint security. Endpoints MUST ignore DNS_ASSIGN capsules unless it has reason to trust its peer and is expecting DNS configuration from it.

This mechanism can cause an endpoint to use a nameserver that is outside of the connect-ip tunnel. While this is acceptable in some scenarios, in others it could break the privacy properties provided by the tunnel. To avoid this, implementations need to ensure that DNS_ASSIGN capsules are not sent before the corresponding ROUTE_ADVERTISEMENT capsule.

## 6. IANA Considerations

This document, if approved, will request IANA add the following values to the "HTTP Capsule Types" registry maintained at <https://www.iana.org/assignments/masque>.

| Value | Capsule Type |
|---|---|
| 0x1ACE79EC (if this document is approved, this value will be replaced by a smaller one before publication) | DNS_ASSIGN |
| 0x274C0FBC (if this document is approved, this value will be replaced by a smaller one before publication) | PREF64 |

Table 1

All of these new entries use the following values for these fields:

Status: provisional (permanent if this document is approved)
Reference: This document
Change Controller: IETF
Contact: masque@ietf.org
Notes: None

# 7. References

## 7.1. Normative References

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

[DNS] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/rfc/rfc1035>.

[DNS-TERMS] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/rfc/rfc8499>.

[DoH] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/rfc/rfc8484>.

[DoQ] Huitema, C., Dickinson, S., and A. Mankin, "DNS over Dedicated QUIC Connections", RFC 9250, DOI 10.17487/RFC9250, May 2022, <https://www.rfc-editor.org/rfc/rfc9250>.

[DoT] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/rfc/rfc7858>.

[HTTP-DGRAM] Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", RFC 9297, DOI 10.17487/RFC9297, August 2022, <https://www.rfc-editor.org/rfc/rfc9297>.

[IDNA] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/rfc/rfc5890>.

[IPv6-TRANSLATOR] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <https://www.rfc-editor.org/rfc/rfc6052>.

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[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>.

[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] 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>.

[SVCB-DNS] Schwartz, B., "Service Binding Mapping for DNS Servers", RFC 9461, DOI 10.17487/RFC9461, November 2023, <https://www.rfc-editor.org/rfc/rfc9461>.

### 7.2. Informative References

[IKEv2] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/rfc/rfc7296>.

[IKEv2-DNS] 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>.

[IKEv2-SVCB] Boucadair, M., Reddy.K, T., Wing, D., and V. Smyslov, "Internet Key Exchange Protocol Version 2 (IKEv2) Configuration for Encrypted DNS", RFC 9464, DOI 10.17487/RFC9464, November 2023, <https://www.rfc-editor.org/rfc/rfc9464>.

[PREF64-RA] Colitti, L. and J. Linkova, "Discovering PREF64 in Router Advertisements", RFC 8781, DOI 10.17487/RFC8781, April 2020, <https://www.rfc-editor.org/rfc/rfc8781>.

## Acknowledgments

The mechanism in this document was inspired by [IKEv2], [IKEv2-DNS], and [IKEv2-SVCB]. The authors would like to thank Alex Chernyakhovsky, Tommy Pauly, and other enthusiasts in the MASQUE Working Group for their contributions.

## Authors' Addresses

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States of America
Email: dschinazi.ietf@gmail.com

Yaroslav Rosomakho
Zscaler
120 Holger Way
San Jose, CA 95134
United States of America
Email: yrosomakho@zscaler.com