# DNS-over-HTTPS

## Introduction

Traditional DNS queries and responses are sent over UDP or TCP without encryption. This is vulnerable to eavesdropping and spoofing (including DNS-based Internet filtering). Responses from recursive resolvers to clients are the most vulnerable to undesired or malicious changes, while communications between recursive resolvers and authoritative name servers often incorporate additional protection (https://developers.google.com/speed/public-dns/docs/security#mitigations).

Currently, web-based applications must use browser extensions to take advantage of advanced DNS features such as DANE (https://en.wikipedia.org/wiki/DANE), DNS-SD service discovery (http://www.infoq.com/articles/rest-discovery-dns), or even to look up anything other than IP addresses. Extensions for features that depend on DNSSEC must validate it themselves, as the browser and the OS may not (be able to) validate DNSSEC.

To address these problems, Google Public DNS offers DNSSEC-validating resolution over an encrypted HTTPS connection using a web-friendly API that does not require browser or OS configuration or installing an extension. DNS-over-HTTPS greatly enhances privacy and security between a client and a recursive resolver, and complements DNSSEC to provide end-to-end authenticated DNS lookups.

Google Public DNS provides the API at **`https://dns.google.com/resolve?`** (https://dns.google.com/resolve?name=example) as well as a human-friendly web interface at **`https://dns.google.com/query?`** (https://dns.google.com/query?name=example). Although the latter displays results in a browser it does not implement the API; be aware not to confuse these two URLs.

## API Specification

**Note:** The API is in beta and subject to changes. To report a bug or request a new feature, please submit an issue (//code.google.com/a/google.com/p/public-dns/issues/entry).

All API calls are HTTP GET requests. The parameter names are case-insensitive. In case of duplicate parameters, the first value will be used. The use of **`http:`** URLs for API calls is blocked with 403 Forbidden.

### Supported parameters

**name**

string, required

The only required parameter. Its length must be between 1 and 253 (https://stackoverflow.com/a/28918017/18829) (ignoring an optional trailing dot if present). All labels (parts of the name separated by dots) must be 1 to 63 bytes long. The API does not support names with escaped or non-ASCII characters, but they are not explicitly rejected. Internationalized domain names must use punycode format (e.g. "xn--qxam" rather than "ελ").

**type**

string, default: `1`

RR type can be represented as a number in [1, 65535] or a canonical string (case-insensitive, such as `A` or `aaaa`). You can use `255` for 'ANY' queries but be aware that this is *not* a replacement for sending queries for both A and AAAA or MX records. Authoritative name servers need not return all records for such queries; some do not respond, and others (such as cloudflare.com) return only HINFO.

**cd**

boolean, default: `false`

The CD (checking disabled) bit. Use `cd`, `cd=1`, or `cd=true` to disable DNSSEC validation; use `cd=0`, `cd=false`, or no `cd` parameter to enable DNSSEC validation.

**edns_client_subnet**

string, default: empty

The edns0-client-subnet option. Format is an IP address with a subnet mask. Examples: `1.2.3.4/24`, `2001:700:300::/48`.

If you are using DNS-over-HTTPS because of privacy concerns, and do not want *any* part of your IP address to be sent to authoritative name servers for geographic location accuracy, use `edns_client_subnet=0.0.0.0/0`. Google Public DNS normally sends approximate network information (usually zeroing out the last part of your IPv4 address).

**random_padding**

string, ignored

The value of this parameter is ignored. Example: `XmkMw-o_mgP2pf.gpw-Oi5dK`.

API clients concerned about possible side-channel privacy attacks using the packet sizes of HTTPS GET requests can use this to make all requests exactly the same size by padding requests with random data. To prevent misinterpretation of the URL, restrict the padding characters to the unreserved URL characters (http://stackoverflow.com/a/695469/18829): upper- and lower-case letters, digits, hyphen, period, underscore and tilde.

### DNS response in JSON

A successful response (comments added here are not present in actual responses):

```
{
  "Status": 0,   // NOERROR - Standard DNS response code (32 bit integer).
  "TC": false,   // Whether the response is truncated
  "RD": true,    // Always true for Google Public DNS
  "RA": true,    // Always true for Google Public DNS
  "AD": false,   // Whether all response data was validated with DNSSEC
  "CD": false,   // Whether the client asked to disable DNSSEC
  "Question":
  [
    {
      "name": "apple.com.",  // FQDN with trailing dot
      "type": 1              // A - Standard DNS RR type
    }
  ],
  "Answer":
  [
    {
      "name": "apple.com.",   // Always matches name in the Question section
      "type": 1,              // A - Standard DNS RR type
      "TTL": 3599,            // Record's time-to-live in seconds
      "data": "17.178.96.59"  // Data for A - IP address as text
    },
    {
      "name": "apple.com.",
      "type": 1,
      "TTL": 3599,
      "data": "17.172.224.47"
    },
    {
      "name": "apple.com.",
      "type": 1,
      "TTL": 3599,
      "data": "17.142.160.59"
    }
  ],
  "Additional": [ ],
  "edns_client_subnet": "12.34.56.78/0"  // IP address / scope prefix-length
}
```

See the EDNS Client Subnet RFC draft (https://tools.ietf.org/html/draft-ietf-dnsop-edns-client-subnet-08#section-6) for details about "scope prefix-length" and how it affects caching.

A failure response with diagnostic information:

```
{
  "Status": 2,  // SERVFAIL - Standard DNS response code (32 bit integer).
  "TC": false,  // Whether the response is truncated
  "RD": true,   // Always true for Google Public DNS
  "RA": true,   // Always true for Google Public DNS
  "AD": false,  // Whether all response data was validated with DNSSEC
  "CD": false,  // Whether the client asked to disable DNSSEC
  "Question":
  [
    {
      "name": "dnssec-failed.org.",  // FQDN with trailing dot
      "type": 1                      // A - Standard DNS RR type
    }
  ],
  "Comment": "DNSSEC validation failure. Please check http://dnsviz.net/d/dnssec-failed.org/dnssec/."
}
```

SPF and TXT records with embedded quotes and name server attribution:

```
{
  "Status": 0,  // NOERROR - Standard DNS response code (32 bit integer).
  "TC": false,  // Whether the response is truncated
  "RD": true,   // Always true for Google Public DNS
  "RA": true,   // Always true for Google Public DNS
  "AD": false,  // Whether all response data was validated with DNSSEC
  "CD": false,  // Whether the client asked to disable DNSSEC
  "Question": [
    {
      "name": "e.dns-example.info.",  // FQDN with trailing dot
      "type": 99                      // SPF - Standard DNS RR type
    }
  ],
  "Answer": [
    {
      "name": "e.dns-example.info.",   // Always matches name in Question
      "type": 99,                      // SPF - Standard DNS RR type
      "TTL": 21599,                    // Record's time-to-live in seconds
      "data": "\"v=spf1 -all\""        // Data for SPF - quoted string
    }
  ],
  "Comment": "Response from 216.239.38.110"
  // Uncached responses are attributed to the authoritative name server
}

{
  "Status": 0,  // NOERROR - Standard DNS response code (32 bit integer).
  "TC": false,  // Whether the response is truncated
  "RD": true,   // Always true for Google Public DNS
  "RA": true,   // Always true for Google Public DNS
  "AD": false,  // Whether all response data was validated with DNSSEC
  "CD": false,  // Whether the client asked to disable DNSSEC
  "Question": [
    {
      "name": "s1024._domainkey.yahoo.com.", // FQDN with trailing dot
      "type": 16                             // TXT - Standard DNS RR type
    }
  ],
  "Answer": [
    {
      "name": "s1024._domainkey.yahoo.com.", // Always matches Question name
      "type": 16,                            // TXT - Standard DNS RR type
      "data": "\"k=rsa;  p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDrEee0RI4Juz+QfiWYui/E9UGSXau/2P8LjnTD8V4Unn+2FAZVGE3kL23bzeoULYv4PeleB3gfm\"\"JiDJOKUJ
      // Data for TXT - multiple quoted strings
    }
  ],
}
```

"Master file" format for string RRs (TXT and SPF) requires double-quotes, and these are escaped within the JSON double-quoted strings. Individual strings are limited to 255 bytes, so longer TXT records such as the yahoo.com DKIM record have multiple strings, each one quoted separately. Many uses of longer TXT record formats such as RFC 4408 (SPF) (https://tools.ietf.org/html/rfc4408#section-3.1.3) or RFC 4871 (DKIM) (https://tools.ietf.org/html/rfc4871#section-3.6.2.2) require treating multiple strings in a TXT or SPF record as if they were concatenated but this is not required in general.

**Note:** Although HTTP does not truncate, we may get a truncated response from an authoritative name server. Thus we need the TC bit in the response.

**Note:** Given that there is no size limit in HTTP message, we do not need the general EDNS mechanism. The only meaningful EDNS option (edns-client-subnet) is a parameter in the request and a top level field in the JSON response.

*Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/), and code samples are licensed under the Apache 2.0 License (http://www.apache.org/licenses/LICENSE-2.0). For details, see our Site Policies (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.*

*Last updated July 21, 2016.*