



DomainTools Farsight DNSDB API

Generated on April 21, 2026

Contents

1	What is DNSDB?	5
2	API Versions	6
3	Key Differences in API v2	6
4	Quick Start	6
4.1	1. Set Up Authentication	6
4.2	2. Check Your Quota	6
4.3	3. Make Your First Query	6
5	Core Concepts	7
5.1	Request Types	7
5.2	Query Parameters	7
5.3	Common Query Patterns	7
5.3.1	Name -> Answers	7
5.3.2	IP -> Names	7
5.3.3	Wildcard Searches	7
6	Understanding Results	8
6.1	Streaming API Framing (SAF)	8
6.2	Result Metadata	8
7	Next Steps	8
8	Getting Help	8
9	DNSDB API authentication	9
9.1	API Key Authentication	9
9.1.1	Example	9
9.1.2	Authentication Errors	9
9.2	Name Encoding	9
9.3	Setting Up Your Environment	9
10	DNSDB service limits and quotas	10

10.1 Concurrent Connections	10
10.2 Quota Types	10
10.2.1 Time-Based Quotas	10
10.2.2 Block-Based Quotas	10
10.2.3 Unlimited Quotas	10
10.3 Burst Rate Quota	10
10.4 Checking Your Quota	11
10.4.1 Rate Limit Response	11
10.5 Examples	11
10.5.1 Time-Based Quota Example	11
10.5.2 Block-Based Quota Example	12
10.5.3 Unlimited Quota Example	12
10.6 X-RateLimit Headers	12
10.6.1 Time-Based Quota Headers	12
10.6.2 Block-Based Quota Headers	12
10.6.3 Unlimited Quota Headers	13
10.7 Quota Exceeded Errors	13
11 DNSDB API HTTP response codes	14
11.1 Success Codes	14
11.2 Client Error Codes	14
11.3 Server Error Codes	15
11.4 Content Types by Response	15
11.5 Examples	15
11.5.1 Successful Query	15
11.5.2 Bad Request	16
11.5.3 Rate Limit Exceeded	16
12 DNSDB API User Guide	17
12.1 Introduction	17
12.2 DNSDB Capabilities and Limits	17
12.2.1 Trial Products	17
12.2.2 Subscription Products	17
12.3 User Guide Notes	17
12.4 Primary Pivots	18
12.4.1 Name -> Answers	18
12.4.2 Wildcard Left Hand Side	18
12.4.3 Wildcard Right Hand Side	18
12.4.4 IP -> Names	18
12.4.5 Name -> Names	19
12.5 Considerations	19
13 DNSDB Streaming API Framing Protocol	20
13.1 Introduction	20
13.2 Audience	20
13.3 Format	20
13.4 State Machine	21
13.5 Examples	21
13.5.1 Simple successful example	21
13.5.2 Long-running example with keep-alive messages	22
13.5.3 Limited example	22
13.5.4 Failure example	22
14 DNSDB summarize requests	23
14.1 URL Path Scheme	23

14.2 Query Parameters	23
14.3 Result Format	23
14.3.1 RRset and Rdata Results	23
14.4 Examples	24
14.4.1 Example 1: Basic Summarize	24
14.4.2 Example 2: No Results Found	24
14.4.3 Example 3: Summarize with limit and max_count	24
14.4.4 Example 4: Summarize with only limit	25
14.4.5 Example 5: Summarize with only max_count	25
14.4.6 Example 6: Summarize IP network	25
15 DNSDB RRset lookups	26
15.1 URL Path Scheme	26
15.2 Type Parameter	26
15.3 Wildcards	26
15.4 RRtype Parameter	26
15.5 Bailiwick Parameter	27
15.6 Wildcard Domain Consolidation	27
15.7 Result Metadata	27
15.8 RRSIG Time Format	27
15.9 Result Format	28
15.10 Examples	28
15.10.1 Example 1: Lookup all RRsets for www.farsightsecurity.com	28
15.10.2 Example 2: Lookup all NS RRsets ending in farsightsecurity.com	29
15.10.3 Example 3: Lookup DNSSEC records	29
15.10.4 Example 4: Raw query for records under fsi.io	29
16 DNSDB rdata lookups	31
16.1 URL Path Scheme	31
16.2 Type Parameter	31
16.3 RRtype Parameter	31
16.4 IP Address Examples	31
16.5 Result Format	32
16.6 Examples	32
16.6.1 Example 1: Lookup records with IPv4 address 104.244.13.104	32
16.6.2 Example 2: Lookup records in 104.244.13.104/29 network	33
16.6.3 Example 3: Lookup records with IPv6 address	33
16.6.4 Example 4: Lookup records in IPv6 network prefix	33
16.6.5 Example 5: Lookup domains delegated to nameserver	34
16.6.6 Example 6: Lookup domains with mail exchanger	34
16.6.7 Example 7: Raw query for records	34
17 DNSDB query parameters	35
17.1 Path Parameters	35
17.1.1 RRset Lookup Path Parameters	35
17.1.2 Rdata Lookup Path Parameters	36
17.2 Query String Parameters	37
17.2.1 Available Parameter Types	37
17.2.2 Example with Query Parameters	37
17.3 Summarize Queries	37
17.4 Next Steps	37
18 DNSDB time-fencing query parameters	38
18.1 Supported Parameters	38
18.2 Time Format	38

18.3 Examples	38
18.3.1 Absolute Time (Unix Timestamps)	38
18.3.2 Relative Time (Seconds in the Past)	38
18.4 Combining Time Parameters	39
18.4.1 Records Only Observed in 2015	39
18.4.2 Old Records Recently Observed	39
18.5 Important Notes	39
18.6 Common Time Values	39
19 DNSDB other query parameters	41
19.1 Common Parameters	41
19.2 Lookup-Specific Parameter	42
19.3 Summarize-Specific Parameter	42
19.4 Usage Examples	42
19.4.1 Limit results to 100	42
19.4.2 Use unaggregated results	43
19.4.3 Return human-readable timestamps	43
19.4.4 Combine multiple parameters	43
19.4.5 Use offset for pagination	43
20 DNSDB Flex Search API Guide	44
20.1 Introduction	44
20.2 Audience	44
20.3 High level features	44
20.3.1 URL encoding	44
20.3.2 curl issue	45
20.4 Summary of differences between APIv2 and the Flex API	45
20.4.1 Summary of commonalities between APIv2 and the Flex API	45
20.5 Streaming API Framing	46
20.6 Authentication	46
20.7 Getting an API Key	46
20.8 Query URL structure	46
20.9 Optional query parameters	47
20.9.1 Time-fencing query parameters	47
20.9.2 Result formats	51
20.10 HTTP Content-Types	51
20.11 HTTP Response codes	52
20.11.1 Errors after a HTTP 200 (OK) status	53
20.11.2 400 Bad Request errors	53
20.12 Service limits and Quotas	54
20.13 Example client	55
20.14 Example results	55
20.14.1 SAF format compared to dnsdbflex output format	55
20.14.2 Setting DNSDB_API_KEY	55
20.14.32. Search for all rnames whose owner name matches regular expression <code>^fsi\.io\$</code> , limited to 2 rnames. Note: This <i>is</i> right anchored	56
20.14.43. Search for all rdata which matches regular expression <code>^fsi\.io</code> , limited to 2 results	56
20.14.54. Use dnsdbflex to search all rnames matching a complex range glob	57
20.14.65. Request with missing lower-level components	57
21 DNSDBQ (Query Tool) Guide	59
21.1 Introduction	59
21.2 Description	59
21.3 Farsight DNSDB	59

21.4 Options	59
21.5 Timestamp Formats	63
21.6 Examples	63
21.7 ASINFO/CIDR Lookups	65
21.8 Files	65
21.9 Environment	65
21.10 Exit Status	66
22 DNSDB Flexible Search Query Tool Reference	67
22.1 Synopsis	67
22.2 DESCRIPTION	67
22.3 OPTIONS	67
22.4 EXAMPLES	69
22.5 TIME FENCING	69
22.6 FILES	70
22.7 ENVIRONMENT	70
22.8 EXIT STATUS	71
22.8.1 SEE ALSO	71
23 DNSDB search glob guide	72
23.1 Glob Syntax	72
23.1.1 Character Class Syntax	72
23.1.2 Important notes	73
23.2 Examples	74
24 DNSDB Search with Compatible Regular Expressions	76
24.1 Regexp syntax	76
24.2 Character class syntax	77
24.3 Important notes	78
24.3.1 Valid wildcard patterns	78
24.3.2 Invalid wildcard patterns	78
24.4 Examples	79
25 DNSDB GitHub repositories	81
25.1 Core Libraries	81
25.2 Python Extensions	81
25.3 Utilities	81
25.4 All Repositories	81
25.5 Licensing	81

Build our world-class domain intelligence into your own tools.

This guide will help you get started with the DNSDB API v2.

1 What is DNSDB?

DNSDB is a database that stores and indexes both the passive DNS data available via Farsight Security’s Security Information Exchange as well as the authoritative DNS data that various zone operators make available. DNSDB makes it easy to search for individual DNS RRsets and provides additional metadata for search results such as first seen and last seen timestamps as well as the DNS bailiwick associated with an RRset.

2 API Versions

There are two versions of the API:

- **API v1:** Legacy version (see [API v1 Reference](#))
- **API v2:** Current version (documented here)

3 Key Differences in API v2

- Results are encapsulated in [Streaming API Framing \(SAF\) Protocol](#)
- Only “jsonl” output format is supported (specified as ‘application/x-ndjson’)
- All API URLs now start with /dnsdb/v2
- The `rate_limit` request moved to /dnsdb/v2/rate_limit
- Rdata values are always returned as an array
- Improved error handling and status codes

4 Quick Start

4.1 1. Set Up Authentication

Set your API key as an environment variable:

```
export DNSDB_API_KEY="your-api-key-here"
```

See [Authentication](#) for more details.

4.2 2. Check Your Quota

Before making queries, check your available quota:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/rate_limit"
```

See [Rate Limits](#) for more information.

4.3 3. Make Your First Query

Lookup all RRsets for a domain:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?limit=10"
```

5 Core Concepts

5.1 Request Types

DNSDB supports four types of requests:

1. **Lookup** - Primary query to search for individual DNS RRsets
 - **RRset Lookups** - Forward lookups based on owner name
 - **Rdata Lookups** - Inverse lookups based on Rdata values
2. **Summarize** - Returns a summary of RRsets that would be returned by a lookup query
3. **Ping** - End-to-end connectivity test (no API key required)
4. **Rate Limit** - Returns quota information

5.2 Query Parameters

Enhance your queries with optional parameters:

- **Time-Fencing** - Filter results by time ranges
- **Other Parameters** - Control result limits, aggregation, and more

5.3 Common Query Patterns

5.3.1 Name -> Answers

Get all historical answers for a domain:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/www.example.com"
```

5.3.2 IP -> Names

Find all names pointing to an IP:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/104.244.13.104"
```

5.3.3 Wildcard Searches

Enumerate subdomains:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/*.example.com"
```

6 Understanding Results

6.1 Streaming API Framing (SAF)

All lookup and summarize results are wrapped in SAF protocol:

```
{"cond": "begin"}
{"obj": {...result data...}}
{"obj": {...result data...}}
{"cond": "succeeded"}
```

The final cond value indicates success: - "succeeded" - Query completed successfully -
"limited" - Result limit reached - Other values indicate errors or truncation

See [Streaming Protocol](#) for details.

6.2 Result Metadata

Each result includes: - **count**: Number of times observed - **time_first/time_last**: First and last observation timestamps - **rname**: Owner name - **rrtype**: Record type - **rdata**: Record data (always an array) - **bailiwick**: DNS zone context (rrset lookups only)

7 Next Steps

- Review the [User Guide](#) for detailed usage patterns
- Check the [FAQ](#) for common questions
- Explore [CLI Tools](#) for command-line access
- Read about [Flexible Search](#) for advanced queries

8 Getting Help

- Review the [FAQ](#)
- Contact [DomainTools Support](#)
- Request a demo from the [DomainTools sales team](#)

9 DNSDB API authentication

The DNSDB API is provided over an encrypted HTTPS transport over the Internet at the following URL:

<https://api.dnsdb.info/>

9.1 API Key Authentication

Authentication is performed by providing an API key (a long string of hexadecimal digits or dashes) in a special HTTP request header, X-API-Key.

9.1.1 Example

If your API key is d41d8cd98f00b204e9800998ecf8427e, then the following HTTP header should be added to the HTTP request:

```
X-API-Key: d41d8cd98f00b204e9800998ecf8427e
```

9.1.2 Authentication Errors

If the X-API-Key header is not present, or the provided API key is not valid, a 403 Forbidden response will be returned to the HTTP client.

9.2 Name Encoding

All IDN DNS names used with the DNSDB API must be encoded in [Punycode](#).

Warning

Using non-basic ASCII characters may result in a 500 Internal Server error.

9.3 Setting Up Your Environment

For convenience in examples throughout this documentation, you can set your API key as an environment variable:

```
export DNSDB_API_KEY="d41d8cd98f00b204e9800998ecf8427e"
```

Then use it in curl commands:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com"
```

10 DNSDB service limits and quotas

The DNSDB API implements quota management to control usage and ensure fair access to resources.

10.1 Concurrent Connections

The number of concurrent connections to a DNSDB API server may be limited. This limit is separate from the quota limit described below.

Warning

If the concurrent connection limit is exceeded, the HTTP 503 “Service Unavailable” response code will be generated.

10.2 Quota Types

API keys have a primary quota, which limits the number of requests that can be made to the data-fetching endpoints (`lookup`, `/dnsdb/v2/lookup`, `summarize`, and `/dnsdb/v2/summarize`). There are three types of quotas:

10.2.1 Time-Based Quotas

Time-based quotas are usually applied on a daily basis and reset daily at 00:00 (midnight) in the UTC time zone. Time-based quotas can also be applied for arbitrary time-quantum, but this is less common.

10.2.2 Block-Based Quotas

For block quotas, there are a number of queries that are bought, with all, or a subset, of that number “split” off for a particular API key. Block quotas have:

- A specific number of queries split with an expiration time (usually much longer than a day)
- The DNSDB API server tracks the number of queries split, number used, and the expiration time
- The number of queries bought is not visible to the DNSDB API server

Warning

If a block quota is expired, a 401 “Unauthorized” response code will be generated with message ‘Error: Quota is expired’.

10.2.3 Unlimited Quotas

Unlimited quotas do not limit the number of queries per day.

10.3 Burst Rate Quota

There may also be a secondary, burst rate, quota associated with an API key. The burst rate limits how many requests may be made in a short time window.

Example: 5 requests in 360 seconds

The parameters for burst rate are:

- **burst_size**: Maximum number of requests in the window
- **burst_window**: Time window in seconds

10.4 Checking Your Quota

You may query the `/dnsdb/v2/rate_limit` endpoint to obtain a JSON map containing quota information. Querying the `/dnsdb/v2/rate_limit` endpoint does not count against the quota limit.

10.4.1 Rate Limit Response

The `/dnsdb/v2/rate_limit` endpoint returns a JSON map named `rate` containing some or all of the following:

Key	Description
<code>limit</code>	The maximum number of API lookups that may be performed. This is the initial quota.
<code>remaining</code>	For time-based quotas: the remaining number of API lookups that may be performed until the reset time. For block-based quotas: the remaining number of API lookups in the block quota.
<code>reset</code>	For time-based quotas: UNIX epoch timestamp with second granularity indicating the next point in time when the quota limit will be reset. Usually this is at 00:00 (midnight) UTC. For block-based quotas: the value will be "n/a".
<code>expires</code>	Only present for block-based quota: UNIX epoch timestamp with second granularity indicating when the quota will expire.
<code>results_max</code>	Returns the maximum number of results that can be returned by these lookup methods. This overrides a "limit" query parameter if provided. For example, if "?limit=20000" is appended to the URL path but <code>results_max=1000</code> then only up to 1000 results will be returned.
<code>offset_max</code>	The maximum value that the <code>offset</code> query parameter can be. If it is higher then an HTTP 416 "Requested Range Not Satisfiable" response code will be returned with message "Error: offset value greater than maximum allowed." If the value is "n/a" then the <code>offset</code> parameter is not allowed for this API key, and similar 416 error will be generated.
<code>burst_size</code>	The maximum number of API lookups that may be performed within this <code>burst_window</code> number of seconds.
<code>burst_window</code>	The number of seconds over which a burst of queries is measured.

Note

If `burst_size` and `burst_window` are not returned then there is no burst rate quota applicable.

10.5 Examples

10.5.1 Time-Based Quota Example

For a time-based quota, the following is an example of a `/dnsdb/v2/rate_limit` response performed on June 10, 2015 that indicates that the API key's quota limit will be reset at

midnight UTC on June 11, 2015, and that 999 lookups are remaining out of a total quota of 1000 lookups:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/rate_limit"
```

Response:

```
{"rate": {"reset": 1433980800, "limit": 1000, "remaining": 999}}
```

10.5.2 Block-Based Quota Example

For a block-based quota, the following example indicates that the API key's quota limit will expire at Mon Apr 15 19:28:34 2019, 592 lookups were used out of a total quota of 600 lookups, the burst size is 10 queries in 5 minutes, the maximum number of results that can be requested from a single query are 256, and *offset* will fail if higher than 3000000:

```
{"rate": {"reset": "n/a", "burst_size": 10, "expires": 1555370914, "burst_window":
↪ 300,
  "offset_max": 3000000, "results_max": 256, "limit": 600, "remaining": 8}}
```

10.5.3 Unlimited Quota Example

An unlimited quota API key has the corresponding `/dnsdb/v2/rate_limit` response:

```
{"rate": {"reset": "n/a", "limit": "unlimited", "remaining": "n/a"}}
```

10.6 X-RateLimit Headers

Responses from the data-fetching endpoints contain information in the HTTP response headers that are a subset of that obtained from the `/dnsdb/v2/rate_limit` endpoint. These values are embedded as the `X-RateLimit-Limit`, `X-RateLimit-Remaining`, `X-RateLimit-Reset`, and for block-based quotas, `X-RateLimit-Expires` headers.

10.6.1 Time-Based Quota Headers

For a time-based quota, responses to lookups will contain response headers like:

```
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 999
X-RateLimit-Reset: 1433980800
```

10.6.2 Block-Based Quota Headers

For a block-based quota, responses to lookups will contain response headers like:

```
X-RateLimit-Limit: 600
X-RateLimit-Remaining: 8
```

```
X-RateLimit-Reset: n/a  
X-RateLimit-Expires: 1555370914
```

10.6.3 Unlimited Quota Headers

For an unlimited API key, responses to lookups will contain response headers like:

```
X-RateLimit-Limit: unlimited  
X-RateLimit-Remaining: n/a  
X-RateLimit-Reset: n/a
```

10.7 Quota Exceeded Errors

If you have exceeded your quota, the HTTP 429 “Too Many Requests” response code will be generated with message ‘Error: Rate limit exceeded’.

- **For time-based quotas:** The API key’s daily quota limit is exceeded. The quota will automatically replenish, usually at the start of the next day.
- **For block-based quotas:** The block quota is exhausted. You may need to purchase a larger quota.
- **For burst rate secondary quotas:** There were too many queries within the burst window. The window will automatically reopen at its end.

11 DNSDB API HTTP response codes

The DNSDB API uses standard HTTP response codes to indicate the success or failure of requests.

11.1 Success Codes

Response Code	Description
200 OK	The request was successfully processed. For requests that stream back data, the server interpreted the request correctly and began to reply, but look at the final "cond" for success or failure of returning the streamed data.

11.2 Client Error Codes

Response Code	Description
400 Bad Request	The URL is formatted incorrectly, such as unrecognized lower-level components.
401 Unauthorized	Your API key may not query this API version, or the API key is not authorized (usually indicates the block quota is expired).
403 Forbidden	The X-API-Key header is not present, the provided API key is not valid, or the Client IP address is not authorized for this API key.
404 Not Found	The URL path had unrecognized or missing top-level components. Note: In APIv1 this error indicated that there were no results found. In APIv2, this is not an error - a 200 OK is returned with no "obj" lines in the SAF response.
415 Unsupported Media Type	The Accept: header does not specify a supported content type for this query.
416 Requested Range Not Satisfiable	The offset value is greater than the maximum allowed or if an offset value was provided when not permitted.

Response Code	Description
429 Too Many Requests	You have exceeded your quota and no new requests will be accepted at this time. For time-based quotas: The API key's daily quota limit is exceeded. The quota will automatically replenish, usually at the start of the next day. For block-based quotas: The block quota is exhausted. You may need to purchase a larger quota. For burst rate secondary quotas: There were too many queries within the burst window. The window will automatically reopen at its end.

11.3 Server Error Codes

Response Code	Description
500 Internal Server Error	There is an error processing the request. Returns text/html content type regardless of what was requested.
503 Service Unavailable	The limit of number of concurrent connections is exceeded.
504 Gateway Timeout	The DNSDB Server timed-out before returning any results. Your query might be too complex, such as doing a wildcard search with restrictive time-fencing. You might simplify the query and retry it.

11.4 Content Types by Response

Different response types return different content types:

- **Successful data responses (200 OK):** Returns the content type specified in the Accept header (e.g., `application/x-ndjson`)
- **500 Internal Server Error:** Returns `text/html` (no matter what was requested)
- **Other API errors:** Return `text/plain` (no matter what was requested)
- **nginx errors** (e.g., 404 for unknown URL): Return `text/html`

11.5 Examples

11.5.1 Successful Query

```
curl -i -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?limit=1"
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/x-ndjson
X-RateLimit-Limit: 1000
X-RateLimit-Remaining: 999
X-RateLimit-Reset: 1433980800

{"cond": "begin"}
{"obj": {...}}
{"cond": "succeeded"}
```

11.5.2 Bad Request

```
curl -i -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/name/fsi.io"
```

Response:

```
HTTP/1.1 400 Bad Request
Content-Type: text/plain

Error: unable to parse request
```

11.5.3 Rate Limit Exceeded

```
curl -i -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com"
```

Response:

```
HTTP/1.1 429 Too Many Requests
Content-Type: text/plain

Error: Rate limit exceeded
```

12 DNSDB API User Guide

12.1 Introduction

DNSDB is a database that stores and indexes both passive DNS data (from Farsight Security's Security Information Exchange, SIE) and authoritative DNS data provided by various zone operators. It allows searching for individual DNS RRsets, with additional metadata for search results, such as first seen and last seen timestamps and the DNS bailiwick for each RRset. DNSDB also supports inverse (rdata) searches.

12.2 DNSDB Capabilities and Limits

Access to DNSDB is licensed in several ways, with different interfaces and tools providing various capabilities and limits. Understand your license and toolset for quota and feature differences.

12.2.1 Trial Products

Product	Quota	Maximum Results	Duration	Data Available	Rate Limit	Query Privacy
Maltego Free Queries	12/hour	12	N/A	2010 to now	12/hour	No

To request a DNSDB demonstration, contact the [DomainTools sales team](#).

12.2.2 Subscription Products

Product	Quota	Maximum Results	Duration	Data Available	Rate Limit	Query Privacy
Queries per Day	1K – Unlimited	10K – 1M	1 Year	2010 to now	None	Yes

12.3 User Guide Notes

- In Curl examples, \$APIKEY is an environment variable. Set it in the current shell with (example only):

```
APIKEY="QmIodGqF12TK0f8bqBe6S6WxvZ4LTtzP1VLS09g0UApw28gedka5450cumVW4wHkB"
```

- API calls below use API Version 2.
- Curl 7.42.0+ supports the --path-as-is option, preventing curl from merging/squashing ../ or ./ sequences.

12.4 Primary Pivots

When investigating historical DNS data, five primary pivots are useful:

1. Name -> Answers (names and IPs)
2. Wildcard left hand side
3. Wildcard right hand side
4. IP -> Names
5. Name -> Names

12.4.1 Name -> Answers

Specify a name to retrieve historical answers (A, AAAA, NS, MX, SOA, TXT, etc.). If no record type is specified, all are returned.

12.4.1.1 Hostname Example

- Hostname: `www.fsi.io`

```
curl -s -H 'Accept: application/x-ndjson' -H "X-API-Key: $APIKEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/www.fsi.io?limit=10000"
```

12.4.1.2 Second Level Domain Example

- Hostname: `fsi.io`

```
curl -s -H 'Accept: application/x-ndjson' -H "X-API-Key: $APIKEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/fsi.io?limit=10000"
```

12.4.2 Wildcard Left Hand Side

Enumerate all subdomains of a second-level domain.

```
curl -s -H 'Accept: application/x-ndjson' -H "X-API-Key: $APIKEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/*.fsi.io?limit=10000"
```

12.4.3 Wildcard Right Hand Side

Search for a base domain or TLD with a wildcard on the right.

```
curl -s -H 'Accept: application/x-ndjson' -H "X-API-Key: $APIKEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.*?limit=10000"
```

12.4.4 IP -> Names

Return any names pointing to a specific IP.

```
curl -s -H 'Accept: application/x-ndjson' -H "X-API-Key: $APIKEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/104.244.13.0,24?limit=10000"
```

12.4.5 Name -> Names

Return any names pointing to a name (e.g., NS records).

```
curl -s -H 'Accept: application/x-ndjson' -H "X-API-Key: $APIKEY" \  
↳ "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/name/ns1.infocity.club/NS?limit=10000"
```

12.5 Considerations

1. If you plan to truncate displayed answers, make a larger query to the API (limit $\geq 10,000$) to get recent results.
2. API data is not sorted. For investigations, sort by `time_last` to get the most recent records.
3. SOA records may contain email addresses—useful for pivoting into other data sources (e.g., historical WHOIS).
4. In a UI, allow hyperlink pivoting between names and IPs for better UX.
5. Enable filtering of DNS record types to parse results more easily.

13 DNSDB Streaming API Framing Protocol

13.1 Introduction

This document describes the Farsight Streaming API Framing (SAF) protocol, a data transfer protocol. It is initially used by DNSDB to transfer pre-standard IETF COF format JSON objects. SAF is basically a wrapper protocol around JSON objects.

13.2 Audience

This document is intended for programmers who want to write applications that can interact with the RESTful DNSDB API using JSON and HTTP.

13.3 Format

The data is transmitted as newline delimited JSON lines, aka JSONL or NDJSON.

All string values are UTF-8. All attribute names are US-ASCII.

Whitespace within the JSONL structure should be ignored.

Each JSONL object contains the optional attributes:

1. “cond” is a condition. It is a scalar enum of string values:
 - “begin”
 - “ongoing”
 - “succeeded”
 - “limited”
 - “failed”
2. “msg” is a human readable message. It is a scalar string which ought to be reported to the end-user or in the system log.
3. “obj” contains data appropriate for the protocol using SAF encapsulation. It is a wrapped JSON object. For the DNSDB API, it contains a pre-standard IETF COF object.

Other attributes, if present, would be the subject of some future protocol revision.

The condition “begin” MUST be in the first, initiating JSONL object. This object MUST contain a “cond” and MAY contain a “msg”.

The conditions “succeeded”, “limited”, and “failed” are terminating conditions and MUST be in the last JSONL object.

JSONL objects after the first and before the last MAY contain an “obj” attribute, MAY contain a “msg” attribute, and MAY either specify “cond” to be “ongoing” OR omit the “cond”. The default value for “cond” is “ongoing”.

If a terminating condition is missing (i.e. the last object in the data stream does not contain one) then the client should interpret this as truncation and generate an appropriate error message to report to the end-user or in the system log. The previously received data objects may be discarded or used subject to the knowledge they may be incomplete. This is also the case if the data stream terminates with a TCP error. A client is free to retry a truncated operation.

If any JSONL object cannot be parsed as valid JSON then both that JSONL object and all remaining JSONL objects must be discarded and some warning must be issued to the user or system log.

If the terminating condition is “succeeded” or “limited” then the data objects are valid.

If the terminating condition is “failed” then the data objects may be discarded or used subject to the knowledge they may be incomplete.

If a JSONL object after the first and before the last, either having no “cond” attribute or having a “cond” value of “ongoing”, contains a “msg” attribute then the “msg” describes an informational or warning condition which has no decisive bearing on the validity of the objects in the response stream. Other values of “msg” SHOULD be reported to the end-user or in the system log.

If a JSONL object after the first and before the last, either having no “cond” attribute or having a “cond” value of “ongoing”, contains neither a “obj” nor a “msg”, then this is a “keep alive” message. A keep alive message is designed to signal to the client that the connection is still open. A legal representation of a keep alive message is the empty JSON object {}. Keep alives are useful for queries that take a long time to process before reporting anything to the user. Keep alives are also known as “heartbeats”.

13.4 State Machine

The state machine grammar is:

```
initiating JSONL object :=
  { "cond": "begin" [, "msg": "$message"] }
```

```
ongoing JSONL object :=
  { [ "cond": "ongoing", ] ["obj": { $object } ] [, "msg": "$message"] }
```

```
terminating JSONL object :=
  { "cond": "succeeded" [, "msg": "$message"] }
  |
  { "cond": "limited" [, "msg": "$message"] }
  |
  { "cond": "failed" [, "msg": "$message"] }
```

START => initiating JSONL object => ongoing JSONL object OR terminating JSONL object

ongoing JSONL object => ongoing JSONL object OR terminating JSONL object

terminating JSONL object => END

\$message => scalar string

\$object => JSON object wrapped by the SAF protocol

13.5 Examples

13.5.1 Simple successful example

```
{"cond": "begin"}
{"obj":{"count":10392,"time_first":138126549...}}
{"cond": "succeeded"}
```

The second JSONL object implicitly has the default condition of “ongoing”.

13.5.2 Long-running example with keep-alive messages

This example shows a query that took a while to return the first response, and then had occasional delays in responding. Because “cond”:“ongoing” is the default, it is normally omitted, which means that a “keep alive” message is transmitted as an empty JSON object, {}. The example also shows varying amounts of whitespace, as the whitespace should be ignored.

```
{"cond":"begin"}
{}
{}
{"obj":{"count":10392,"time_first":138126549...}}
...
{}
{"obj": {"count": 1234,"time_first": 238126549...}}
...
{}
...
{"obj" : {"count" :456, "time_first":338126549...} }
{"cond":"succeeded"}
```

13.5.3 Limited example

```
{"cond": "begin"}
{"obj":{"count":10392,"time_first":138126549...}}
{"obj":{"count":33,"time_first":19126549...}}
{"cond":"limited", "msg":"Result limit reached"}
```

In this case, the results are valid and SHOULD be used.

Note: The “Result limit reached” message does not imply that re-trying the query with a higher limit value will return more results, only that the limit was reached.

13.5.4 Failure example

```
{"cond": "begin"}
{"obj":{"count":33,"time_first":19126549...}}
...
{"cond": "failed", "msg": "Processing timeout; results may be incomplete"}
```

In this case, the results MAY be used, but it is recommended to retry the request.

14 DNSDB summarize requests

Summarize requests return a summary of RRsets that would be returned by a lookup query. This gives you an estimate of result size and provides at-a-glance information on when a given domain name, IP address, or other DNS asset was first-seen and last-seen by the global sensor network, as well as the total observation count.

14.1 URL Path Scheme

All DNSDB summarize requests are rooted in URL paths under the `/dnsdb/v2/summarize` hierarchy. Everything underneath `/dnsdb/v2/summarize` is the same as for `/dnsdb/v2/lookup` queries:

```
/dnsdb/v2/summarize/rrset/name/OWNER_NAME/RRTYPE/BAILIWICK  
/dnsdb/v2/summarize/rdata/TYPE/VALUE/RRTYPE
```

14.2 Query Parameters

Summarize requests accept the same optional query parameters as lookup requests, with one additional parameter:

Parameter	Description
<code>max_count</code>	Controls stopping when the summary count reaches this value. The resulting total count <i>can</i> exceed <code>max_count</code> as it will include the entire count from the last rrsset examined. The default is to not constrain the count. Example: appending “ <code>?max_count=100</code> ” to the URL path will stop the summary when its total count reaches 100.

See [Query Parameters](#) for all other available parameters.

14.3 Result Format

The DNSDB API only supports one Summarize result format, the “json” format (though this should be specified in an HTTP ACCEPT header as ‘application/x-ndjson’).

14.3.1 RRset and Rdata Results

Key	Description
<code>count</code>	The number of times the RRset was observed via passive DNS replication.
<code>num_results</code>	The number of results (RRsets) that would be returned from a Lookup.
<code>time_first, time_last</code>	UNIX epoch timestamps with second granularity indicating the first and last times the RRset was observed via passive DNS replication. Only present if the RRsets were observed via passive DNS replication.

Key	Description
zone_time_first, zone_time_last	UNIX epoch timestamps with second granularity indicating the first and last times the RRset was observed via zone file import. Only present if the RRsets were observed via zone file import.

14.4 Examples

14.4.1 Example 1: Basic Summarize

An example JSON object:

```
{"count":528,"num_results":4,"time_first":1557864746,"time_last":1560524861}
```

As encapsulated by the SAF, this would be expressed on the wire as:

```
{"cond":"begin"}
{"obj":{"count":528,"num_results":4,"time_first":1557864746,"time_
↪ last":1560524861}}
{"cond":"succeeded"}
```

14.4.2 Example 2: No Results Found

If there are no RRsets found by the underlying query, the DNSDB API will return:

```
{"count":0, "num_results":0}
```

As encapsulated by the SAF:

```
{"cond":"begin"}
{"obj":{"count":0,"num_results":0}}
{"cond":"succeeded"}
```

14.4.3 Example 3: Summarize with limit and max_count

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/summarize/rrset/name/www.farsightsecurity.com\
  ?limit=2&max_count=5000"
```

In this call, the summarize processing will limit itself to looking at two underlying results rows and will stop when the count value reaches max_count. Since the first row exceeds max_count, the summarize will only account for the first row.

Response:

```
{"cond":"succeeded","obj":{"count":1127,"num_results":2,"time_first":1557859313,
  "time_last":1560537333}}
```

14.4.4 Example 4: Summarize with only limit

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
↪ "https://api.dnsdb.info/dnsdb/v2/summarize/rrset/name/www.farsightsecurity.com?limit=2"
```

In this call, the summarize processing will limit itself to looking at two underlying results rows. The count is the sum of the counts from the two rows, and the time_first is from the first row while time_last is from the second row.

Response:

```
{"cond": "succeeded", "obj": {"count": 1127, "num_results": 2, "time_first": 1557859313, "time_last": 1560537333}}
```

14.4.5 Example 5: Summarize with only max_count

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
  "https://api.dnsdb.info/dnsdb/v2/summarize/rrset/name/www.farsightsecurity.com\  
  ?max_count=50000"
```

Response:

```
{"cond": "succeeded", "obj": {"count": 1078, "num_results": 2, "time_first": 1573594176, "time_last": 1576187607}}
```

14.4.6 Example 6: Summarize IP network

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
  "https://api.dnsdb.info/dnsdb/v2/summarize/rdata/ip/104.244.13.104,29"
```

Response:

```
{"cond": "succeeded", "obj": {"count": 528, "num_results": 4, "time_first": 1557864746, "time_last": 1560524861}}
```

15 DNSDB RRset lookups

RRset lookups query DNSDB's RRset index, which supports "forward" lookups based on the owner name of an RRset.

15.1 URL Path Scheme

```
/dnsdb/v2/lookup/rrset/TYPE/VALUE/RRTYPE/BAILIWICK
```

Both TYPE and VALUE are required; RRTYPE and BAILIWICK are optional. If RRTYPE is not specified, the query functions as if "ANY" was specified.

15.2 Type Parameter

The TYPE parameter specifies how VALUE is interpreted:

Type	Description
name	The VALUE is a DNS owner name in presentation format or wildcards as described below. We sometimes call this just an rrset search.
raw	The VALUE is an even number of hexadecimal digits specifying a raw octet string.

15.3 Wildcards

Wildcards come in two forms:

- **Left-hand wildcard** (*.example.com): Matches any RRsets whose owner names *end with* the given domain name
- **Right-hand wildcard** (www.example.*): Matches any RRsets whose owner names *start with* the given label(s)

Note

Left-hand wildcard queries are somewhat more expensive and slower than right-hand wildcard queries.

15.4 RRtype Parameter

The OWNER_NAME and BAILIWICK are DNS names specified in DNS presentation format. RRTYPE is specified as a DNS RRtype mnemonic.

The RRtype ANY is modified from its usual meaning:

- A DNSDB lookup for RRtype ANY matches any RRtype *except* DNSSEC-related RRtypes: DS, RRSIG, NSEC, DNSKEY, NSEC3, NSEC3PARAM, DLV, CDS, CDNSKEY, and TA
- The pseudo-mnemonic ANY-DNSSEC returns *only* those DNSSEC-related records
- RRtype ANY may be specified for RRTYPE to perform bailiwick filtering without filtering on a particular RRtype

15.5 Bailiwick Parameter

The BAILIWICK may not be specified with raw queries.

15.6 Wildcard Domain Consolidation

Info: July 2022 Update

As of July 2022, DNSDB was changed to reduce junk wildcard domains. We are gradually rolling out a change to replace multiple wildcarded DNS rnames with a single rname starting with a `_WILDCARD_.` label. No other rname labels contain uppercase letters, so records with this (all upper case) `_WILDCARD_.` were never in DNSDB before. Note that there are existing, real domain names that contain a `_wildcard_.` label (all lower case).

15.7 Result Metadata

The results of an rrset lookup return zero or more RRsets, along with the following metadata for each result:

Item	Description
count	The number of times the RRset was observed via passive DNS replication.
bailiwick	The “bailiwick” of an RRset in DNSDB observed via passive DNS replication is the closest enclosing zone delegated to a nameserver which served the RRset. The “bailiwick” of an RRset in DNSDB observed in a zone file is simply the name of the zone containing the RRset.
first seen	A UTC timestamp with seconds granularity indicating the first time an RRset was seen in the given bailiwick via passive DNS replication.
last seen	A UTC timestamp with seconds granularity indicating the last time an RRset was seen in the given bailiwick via passive DNS replication.
first seen in zone file	A UTC timestamp with seconds granularity indicating the first time an RRset was seen in the given bailiwick via zone file import.
last seen in zone file	A UTC timestamp with seconds granularity indicating the last time an RRset was seen in the given bailiwick via zone file import.

Note

An rrset search result may be missing either the pair of (first seen, last seen) timestamps from passive DNS replication or from zone file import, but at least one pair of timestamps will always be present. This may change if a fundamentally new data source is introduced in the future, but there will always be at least one timestamp pair associated with an RRset.

15.8 RRSIG Time Format

Info

The DNSDB API returns RRSIG records with the expiration and inception times in Unix epoch time format (seconds since 1 January 1970 00:00:00 UTC). RFC 4034

also allows presentation of these times in YYYYMMDDHHmmSS format, which many DNSSEC tools and presentation libraries use. It is always possible to distinguish between these two formats because the YYYYMMDDHHmmSS format will always be exactly 14 digits, while the decimal representation of a 32-bit unsigned integer can never be longer than 10 digits.

15.9 Result Format

Results are returned in JSON Lines format with the following keys:

Key	Description
rrname	The owner name of the RRset in DNS presentation format.
rrtype	The resource record type of the RRset, either using the standard DNS type mnemonic, or an RFC 3597 generic type, i.e. the string TYPE immediately followed by the decimal RRtype number.
rdata	An array of one or more Rdata values. The Rdata values are converted to the standard presentation format based on the rrtype value. If the encoder lacks a type-specific presentation format for the RRset's rrtype, then the RFC 3597 generic Rdata encoding will be used.
bailiwick	The "bailiwick" metadata value described above.
count	The number of times the RRset was observed via passive DNS replication.
time_first, time_last	UNIX epoch timestamps with second granularity indicating the first and last times the RRset was observed via passive DNS replication. Will not be present if the RRset was only observed via zone file import.
zone_time_first, zone_time_last	UNIX epoch timestamps with second granularity indicating the first and last times the RRset was observed via zone file import. Will not be present if the RRset was only observed via passive DNS replication.

15.10 Examples

15.10.1 Example 1: Lookup all RRsets for www.farsightsecurity.com

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
↳ "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/www.farsightsecurity.com?limit=2"
```

Response:

```
{"cond": "begin"}
{"obj":{"count":5059,"time_first":1380139330,"time_last":1427881899,
↳ "rrname":"www.farsightsecurity.com.,"rrtype":"A","bailiwick":"farsightsecurity.com.,"
  "rdata":["66.160.140.81"]}}
{"obj":{"count":17381,"time_first":1427893644,"time_last":1468329272,
↳ "rrname":"www.farsightsecurity.com.,"rrtype":"A","bailiwick":"farsightsecurity.com.,"
```

```
"rdata":["104.244.13.104"]}]}}
{"cond": "limited", "msg": "Result limit reached"}
```

15.10.2 Example 2: Lookup all NS RRsets ending in farsightsecurity.com

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
↳ "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/*.farsightsecurity.com/ns/farsightsecurity.com"
```

Response:

```
{"cond": "begin"}
{"obj":{"count":51,"time_first":1372688083,"time_last":1374023864,
↳ "rrname":"farsightsecurity.com.", "rrtype":"NS", "bailiwick":"farsightsecurity.com.",
  "rdata":["ns.lah1.vix.com.", "ns1.isc-sns.net.", "ns2.isc-sns.com.",
  "ns3.isc-sns.info."]}]}
{"obj":{"count":495241,"time_first":1374096380,"time_last":1468324876,
↳ "rrname":"farsightsecurity.com.", "rrtype":"NS", "bailiwick":"farsightsecurity.com.",
  "rdata":["ns5.dnsmadeeasy.com.", "ns6.dnsmadeeasy.com.", "ns7.dnsmadeeasy.com."]}]}
{"cond": "succeeded"}
```

15.10.3 Example 3: Lookup DNSSEC records

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/*.farsightsecurity.com/ANY-
↳ DNSSEC?limit=2"
```

Response:

```
{"cond": "begin"}
{"obj":{"count":1696,"zone_time_first":1374250920,"zone_time_last":1521734545,
  "rrname":"farsightsecurity.com.", "rrtype":"DS", "bailiwick":"com.",
  "rdata":["60454 5 2
↳ 3672C35CFA8FF14C9C223B84277BD645C0AF54BAD5790375FE797161E4801479"]]}]}
{"obj":{"count":3,"zone_time_first":1374250920,"zone_time_last":1374423636,
  "rrname":"farsightsecurity.com.", "rrtype":"RRSIG", "bailiwick":"com.",
  "rdata":["DS 8 2 86400 1374774350 1374165350 8795 com.
  cu0do+2G0yJpBN5ba2zxiljSzgtTzminrVc3CrsNxQPqc5YVQX4eBwMB
  +kpgSEXPT+DF2D9HwIsPpBDNdJekBpXIRW41Yl7IdZYHySqabn7hgt9M
  mk5KNy9gqCOK/JLRs07LPAm3wvfyYer8e0/7VCTejF9/DMbMGsLLH3xr kBA="]}]}
{"cond": "limited", "msg": "Result limit reached"}
```

15.10.4 Example 4: Raw query for records under fsi.io

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/raw/0366736902696f00?limit=2"
```

Response:

```
{"cond": "begin"}  
{"obj":{"count":10392,"time_first":1381265499,"time_last":1428418529,  
  
↔ "rrname":"fsi.io.,"rrtype":"A","bailiwick":"fsi.io.,"rdata":["66.160.140.76"]}  
{"obj":{"count":69435,"time_first":1428433465,"time_last":1538014110,  
  
↔ "rrname":"fsi.io.,"rrtype":"A","bailiwick":"fsi.io.,"rdata":["104.244.13.104"]}  
{"cond": "limited", "msg": "Result limit reached"}
```

16 DNSDB rdata lookups

Rdata lookups query DNSDB's Rdata index, which supports "inverse" lookups based on Rdata record values. In contrast to rrset lookups, rdata lookups return only individual resource records (not full resource record sets) and lack bailiwick metadata.

Tip

An rrset lookup on the owner name reported via an rdata lookup must be performed to retrieve the full RRset and bailiwick.

16.1 URL Path Scheme

```
/dnsdb/v2/lookup/rdata/TYPE/VALUE/RRTYPE
```

Both TYPE and VALUE are required. The RRTYPE parameter is optional.

16.2 Type Parameter

The TYPE parameter specifies how VALUE is interpreted:

Type	Description
name	The VALUE is a DNS domain name in presentation format, or a left-hand ("example.com") or right-hand ("www.example.") wildcard domain name. Note that left-hand wildcard queries are somewhat more expensive than right-hand wildcard queries.
ip	The VALUE is one of an IPv4 or IPv6 single address, with a prefix length, or with an address range. If a prefix is provided, the delimiter between the network address and prefix length is a single comma (",") character rather than the usual slash ("/") character to avoid clashing with the HTTP URI path name separator.
raw	The VALUE is an even number of hexadecimal digits specifying a raw octet string.

16.3 RRtype Parameter

If RRTYPE is not specified, the query functions as if "ANY" was specified.

- For **name** and **raw** rdata lookups, RRTYPE optionally filters the results by RRtype in the same manner as rrset lookups
- For **ip** rdata lookups, the supported RRTYPE values are A, AAAA, and ANY, which can be used interchangeably: you can send an IPv4 or IPv6 address to either value and the data returned will be based on the IP address sent, not on the RRTYPE value
- Any other RRTYPE value for ip lookups will return an "HTTP 400 Bad Request" response

16.4 IP Address Examples

Some examples of "ip" rdata URLs (noting that colon needs to be expressed as %3A):

```

https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/10.0.0.1/ANY
https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/10.0.0.1-10.1.255.255
https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/10.0.0.1,24/ANY
https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/2620%3A11c%3Af008%3A%3A,126
https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/2620%3A11c%3Af008%3A%3A1-
↪ 2620%3A11c%3Af008%3A%3Aff

```

16.5 Result Format

Results are returned in JSON Lines format with the following keys:

Key	Description
rrname	The owner name of the resource record in DNS presentation format.
rrtype	The resource record type of the resource record, either using the standard DNS type mnemonic, or an RFC 3597 generic type, i.e. the string TYPE immediately followed by the decimal RRtype number.
rdata	The record data value. The Rdata value is converted to the standard presentation format based on the rrtype value. If the encoder lacks a type-specific presentation format for the resource record's type, then the RFC 3597 generic Rdata encoding will be used.
count	The number of times the resource record was observed via passive DNS replication.
time_first, time_last	UNIX epoch timestamps with second granularity indicating the first and last times the resource record was observed via passive DNS replication.
zone_time_first, zone_time_last	UNIX epoch timestamps with second granularity indicating the first and last times the resource record was observed via zone file import.

16.6 Examples

16.6.1 Example 1: Lookup records with IPv4 address 104.244.13.104

```

curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/104.244.13.104"

```

Response:

```

{"cond": "begin"}
{"obj":{"count":24,"time_first":1433550785,"time_last":1468312116,
  "rrname":"www.farsighsecurity.com.", "rrtype":"A", "rdata":"104.244.13.104"}}
{"obj":{"count":9429,"time_first":1427897872,"time_last":1468333042,
  "rrname":"farsightsecurity.com.", "rrtype":"A", "rdata":"104.244.13.104"}}
{"cond": "succeeded"}

```

16.6.2 Example 2: Lookup records in 104.244.13.104/29 network

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/104.244.13.104,29"
```

Response (with keep-alive messages):

```
{"cond": "begin"}
{}
{"obj":{"count":24,"time_first":1433550785,"time_last":1468312116,
  "rrname":"www.farsighsecurity.com.,"rrtype":"A","rdata":"104.244.13.104"}}
{"obj":{"count":9429,"time_first":1427897872,"time_last":1468333042,
  "rrname":"farsightsecurity.com.,"rrtype":"A","rdata":"104.244.13.104"}}
{}
{"cond": "succeeded"}
```

16.6.3 Example 3: Lookup records with IPv6 address

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/2620:11c:f004::104"
```

Response:

```
{"cond": "begin"}
{"obj":{"count":14,"time_first":1433845806,"time_last":1467828872,
  ↪ "rrname":"www.farsighsecurity.com.,"rrtype":"AAAA","rdata":"2620:11c:f004::104"}}
{"obj":{"count":5307,"time_first":1427897876,"time_last":1468333042,
  "rrname":"farsightsecurity.com.,"rrtype":"AAAA","rdata":"2620:11c:f004::104"}}
{"cond": "succeeded"}
```

16.6.4 Example 4: Lookup records in IPv6 network prefix

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/ip/2620:11c:f000::,126"
```

Response:

```
{"cond": "begin"}
{"obj":{"count":2,"time_first":1574082633,"time_last":1574082633,
  "rrname":"gw.fmt1.fsi.io.,"rrtype":"AAAA","rdata":["2620:11c:f000::1"]}}
{"obj":{"count":261,"time_first":1573589461,"time_last":1576188661,
  "rrname":"r1.fmt1.fsi.io.,"rrtype":"AAAA","rdata":["2620:11c:f000::2"]}}
{"obj":{"count":241,"time_first":1573611061,"time_last":1576188661,
  "rrname":"r2.fmt1.fsi.io.,"rrtype":"AAAA","rdata":["2620:11c:f000::3"]}}
{"cond": "succeeded"}
```

16.6.5 Example 5: Lookup domains delegated to nameserver

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/name/ns5.dnsmadeeasy.com"
```

Response:

```
{"cond": "begin"}
{"obj":{"count":1078,"zone_time_first":1374250920,"zone_time_last":1468253883,
  "rrname":"farsightsecurity.com.", "rrtype":"NS", "rdata":"ns5.dnsmadeeasy.com."}}
{"obj":{"count":706617,"time_first":1374096380,"time_last":1468334926,
  "rrname":"farsightsecurity.com.", "rrtype":"NS", "rdata":"ns5.dnsmadeeasy.com."}}
{"cond": "succeeded"}
```

16.6.6 Example 6: Lookup domains with mail exchanger

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/name/hq.fsi.io"
```

Response:

```
{"cond": "begin"}
{"obj":{"count":45644,"time_first":1372706073,"time_last":1468330740,
  "rrname":"fsi.io.", "rrtype":"MX", "rdata":"10 hq.fsi.io."}}
{"obj":{"count":19304,"time_first":1374098929,"time_last":1468333042,
  "rrname":"farsightsecurity.com.", "rrtype":"MX", "rdata":"10 hq.fsi.io."}}
{"cond": "succeeded"}
```

16.6.7 Example 7: Raw query for records

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rdata/raw/0366736902696f00?limit=2"
```

Response:

```
{"cond": "begin"}
{"obj":{"count":6,"time_first":1413228451,"time_last":1413228451,
  "rrname":"local-data.fsi.io.", "rrtype":"SOA",
  "rdata":"fsi.io. hostmaster.fsi.io. 2014050101 7200 3600 604800 3600"}}
{"obj":{"count":25,"time_first":1412912798,"time_
↳ last":1412942807,"rrname":"dnstap.info.",
  "rrtype":"SOA", "rdata":"fsi.io. hostmaster.fsi.io. 2014052824 7200 3600
↳ 25920000 3600"}}
{"cond": "limited", "msg": "Result limit reached"}
```

17 DNSDB query parameters

DNSDB API queries use two types of parameters to control searches:

1. **Path Parameters** - Required components that form the core query structure
2. **Query String Parameters** - Optional filters that refine results

17.1 Path Parameters

Path parameters are embedded in the URL path and define what you're searching for. They vary by lookup method.

17.1.1 RRset Lookup Path Parameters

RRset lookups follow this URL structure:

```
/dnsdb/v2/lookup/rrset/TYPE/VALUE/RRTYPE/BAILIWICK
```

Parameter	Required	Description
TYPE	Yes	How VALUE is interpreted: name (DNS owner name or wildcard) or raw (hex octet string)
VALUE	Yes	The search value - interpretation depends on TYPE
RRTYPE	No	DNS record type (A, NS, MX, etc.). If omitted, functions as "ANY"
BAILIWICK	No	Zone context for filtering results. Cannot be used with raw queries

17.1.1.1 TYPE Parameter Values

Type	Description
name	VALUE is a DNS owner name in presentation format or wildcards (*.example.com or www.example.*). Left-hand wildcard queries are more expensive than right-hand.
raw	VALUE is an even number of hexadecimal digits specifying a raw octet string

17.1.1.2 RRTYPE Special Values

- **ANY** - Matches any RRtype *except* DNSSEC-related types (DS, RRSIG, NSEC, DNSKEY, NSEC3, NSEC3PARAM, DLV, CDS, CDNSKEY, TA)
- **ANY-DNSSEC** - Returns *only* DNSSEC-related record types
- If omitted, functions as if "ANY" was specified

17.1.1.3 Examples

```
# Lookup all RRsets for a domain
/dnsdb/v2/lookup/rrset/name/example.com

# Lookup only A records
/dnsdb/v2/lookup/rrset/name/example.com/A

# Wildcard search for subdomains
/dnsdb/v2/lookup/rrset/name/*.example.com/NS/example.com

# Raw query (hex for "fsi.io")
/dnsdb/v2/lookup/rrset/raw/0366736902696f00
```

17.1.2 Rdata Lookup Path Parameters

Rdata lookups follow this URL structure:

```
/dnsdb/v2/lookup/rdata/TYPE/VALUE/RRTYPE
```

Parameter	Required	Description
TYPE	Yes	How VALUE is interpreted: name, ip, or raw
VALUE	Yes	The search value - interpretation depends on TYPE
RRTYPE	No	DNS record type filter. If omitted, functions as "ANY"

17.1.2.1 TYPE Parameter Values

Type	Description
name	VALUE is a DNS domain name in presentation format, or left-hand (.example.com) or right-hand (www.example.) wildcard
ip	VALUE is IPv4/IPv6 address, with prefix length, or address range. Use comma (,) instead of slash (/) for prefix delimiter
raw	VALUE is an even number of hexadecimal digits specifying a raw octet string

17.1.2.2 IP Address Formats

For ip type queries, VALUE can be:

- Single address: 10.0.0.1
- Network prefix: 10.0.0.1,24 (note comma, not slash)
- Address range: 10.0.0.1-10.1.255.255
- IPv6 (URL-encoded): 2620%3A11c%3Af008%3A%3A,126

Note

For ip lookups, RRTYPE values A, AAAA, and ANY are interchangeable - results are based on the IP address provided, not the RRTYPE value.

17.1.2.3 Examples

```
# Find all names pointing to an IP
/dnsdb/v2/lookup/rdata/ip/104.244.13.104

# Find names in a network
/dnsdb/v2/lookup/rdata/ip/104.244.13.104,29

# Find domains using a nameserver
/dnsdb/v2/lookup/rdata/name/ns5.dnsmadeeasy.com

# Find domains with specific mail server
/dnsdb/v2/lookup/rdata/name/mail.example.com
```

17.2 Query String Parameters

Query string parameters are appended to the URL with ? and & to filter and control results. These are optional and work with both rrset and rdata lookups.

17.2.1 Available Parameter Types

- [Time-Fencing Parameters](#) - Filter results by first seen/last seen timestamps
- [Other Parameters](#) - Control result limits, aggregation, formatting, and more

17.2.2 Example with Query Parameters

```
# Combine path and query string parameters
/dnsdb/v2/lookup/rrset/name/example.com/A?limit=100&time_last_after=1468281600
```

This query: - Uses path parameters: name (TYPE), example.com (VALUE), A (RRTYPE) - Uses query parameters: limit=100, time_last_after=1468281600

17.3 Summarize Queries

Summarize queries use the same path parameter structure but under /dnsdb/v2/summarize:

```
/dnsdb/v2/summarize/rrset/TYPE/VALUE/RRTYPE/BAILIWICK
/dnsdb/v2/summarize/rdata/TYPE/VALUE/RRTYPE
```

See [Summarize](#) for details on summarize-specific behavior.

17.4 Next Steps

- Learn about [Time-Fencing Parameters](#) for temporal filtering
- Explore [Other Parameters](#) for result control
- Review [RRset Lookups](#) for detailed examples
- Review [Rdata Lookups](#) for inverse search examples

18 DNSDB time-fencing query parameters

DNSDB records when a specific DNS record was first and last observed. You can filter results by time using time-fencing query parameters.

18.1 Supported Parameters

Parameter	Description
time_first_before	Provide results before the defined timestamp for when the DNS record was first observed.
time_first_after	Provide results after the defined timestamp for when the DNS record was first observed.
time_last_before	Provide results before the defined timestamp for when the DNS record was last observed.
time_last_after	Provide results after the defined timestamp for when the DNS record was last observed.

18.2 Time Format

Time values must be specified as:

- **Unix/Epoch timestamp:** Integer with seconds granularity (e.g., 1420070400)
- **Relative time:** Negative integer representing seconds in the past (e.g., -31536000 for one year ago)

18.3 Examples

18.3.1 Absolute Time (Unix Timestamps)

Filter for records first observed before January 1, 2015:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?time_first_-
↪ before=1420070400"
```

Filter for records last observed before 2013:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?time_last_-
↪ before=1356998400"
```

18.3.2 Relative Time (Seconds in the Past)

Filter for records first observed within the last year:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?time_first_after=-
↵ 31536000"
```

Filter for records last observed within the last 31 days:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?time_last_after=-
↵ 2678400"
```

18.4 Combining Time Parameters

You can combine multiple time parameters to create specific time ranges.

18.4.1 Records Only Observed in 2015

Combine `time_first_after` and `time_last_before` to get records where both first and last observation occurred in 2015:

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?time_first_-
↵ after=1420070399&time_last_before=1451606400"
```

18.4.2 Old Records Recently Observed

Get records first observed before 2012 but last observed within the last month (long-lived records that are still active):

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?time_first_-
↵ before=1325376000&time_last_after=-2678400"
```

18.5 Important Notes

- The DNSDB API server may have API key-dependent time-fencing restrictions
- These restrictions cannot be expanded with query parameters but can be used to narrow the time ranges
- Query parameters can completely override default time ranges if permitted by your API key
- All timestamps use seconds granularity
- Relative times must be preceded by a minus sign (-)

18.6 Common Time Values

For reference, here are some common relative time values:

Duration	Seconds	Parameter Value
1 hour	3,600	-3600
1 day	86,400	-86400
1 week	604,800	-604800
31 days	2,678,400	-2678400
1 year	31,536,000	-31536000

19 DNSDB other query parameters

In addition to time-fencing parameters, DNSDB supports several other optional query parameters that control result behavior and metadata.

19.1 Common Parameters

The following optional parameters may be present in lookup and summarize query URLs:

Parameter	Description
limit	Limit for the number of results returned via these lookup methods. There is a built-in limit to the number of results that are returned via these lookup methods. The default limit is set at 10,000. This limit can be raised or lowered by setting the “limit” query parameter. There is also a maximum number of results allowed; requesting a limit greater than the maximum will only return the maximum. See results_max below for information on that maximum. If “?limit=0” is used then DNSDB will return the maximum number of results allowed. Obviously, if there are less results for the query than the requested limit, only the actual amount can be returned. Example: appending “?limit=20000” to the URL path will set the response limit to 20,000 results.
swclient	Name of the API client software generating the DNSDB query. Limited to twenty alphanumeric characters. This may be logged by the DNSDB API server. Farsight support can help you debug a new API client using this and the following parameter. There is no default. Example: “?swclient=dnsdbq”.
version	Version number of the API client software generating the DNSDB query. Limited to twenty alphanumeric characters plus dash, underscore, and period. This may be logged by the DNSDB API server. There is no default. Example: “?version=1.1a”.
id	Client software specific identity of the user of the API client. Comprised of an alphanumeric string, a colon, and an alphanumeric string, limited to thirty characters. This may be logged by the DNSDB API server. There is no default. Example: “?id=dnsq:91e6245ad31387”.
aggr	Aggregated results group identical rrses across all time periods and is the classic behavior from querying the DNSDB. This means you could get the total number of times an rrses has been observed, but not when it was observed. Unaggregated results ungroup identical rrses, allowing you to see how the domain name was resolved in the DNS across the full-time range covered in DNSDB (subject to time fencing). This can give a more accurate impression of record request <i>volume across time</i> because it will reveal the distinct timestamps of records whose values are repeated. You can answer questions like, “Was a domain parked for a long time, mostly unused, until it was repurposed for serving malware or relaying spam, but then was abandoned again?” It allows you to see if a record was observed heavily in the last week vs. having been observed constantly for years. This is a boolean value. Use True, the default, for the aggregated results or False for unaggregated results. The value is case insensitive and can be abbreviated. Example: “?aggr=f”.

Parameter	Description
humantime	A boolean value that is True if time values (in <code>time_first</code> , <code>time_last</code> , <code>zone_time_first</code> , <code>zone_time_last</code>) should be returned in human readable (RFC3339 compliant) format or False if Unix-style time values in seconds since the epoch should be returned. False is the classic behavior from querying the DNSDB and is the default value for this option. The value is case insensitive and can be abbreviated. Example: “?humantime=t”.

19.2 Lookup-Specific Parameter

The following optional parameter may be present in a *lookup* query URL:

Parameter	Description
offset	How many rows to offset (e.g. skip) in the results. This implements an incremental result transfer feature, allowing you to view more of the available results for a single query. The rows are offset <i>prior</i> to the <i>limit</i> parameter being applied, therefore <i>offset</i> allows seeing additional results past a <i>limit</i> that matches the maximum number of results. Note that DNSDB recalculates the results for each query and the order of results might <i>not</i> be preserved. Therefore, this capability is <i>not</i> a valid way to walk all results over multiple queries – some results might be missing and some might be duplicated. The actual offset that can be used is limited or for certain API keys, <i>offset</i> is not allowed – see the <code>offset_max</code> <code>rate_limit</code> key below. The <i>offset</i> value must be a positive integer. The default is 0, which means do not offset the rows. Example to return the second million results (assuming <code>results_max</code> is 100000): “?limit=100000&offset=100000”.

19.3 Summarize-Specific Parameter

The following optional parameter may be present in a *summarize* query URL:

Parameter	Description
max_count	<i>max_count</i> controls stopping when we reach that summary count. The resulting total count <i>can</i> exceed <i>max_count</i> as it will include the entire count from the last rreset examined. The default is to not constrain the count. Example: appending “?max_count=100” to the URL path will stop the summary when its total count reaches 100.

19.4 Usage Examples

19.4.1 Limit results to 100

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?limit=100"
```

19.4.2 Use unaggregated results

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?aggr=false"
```

19.4.3 Return human-readable timestamps

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?humantime=true"
```

19.4.4 Combine multiple parameters

```
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com\  
  ?limit=1000&aggr=f&humantime=t&swclient=myclient&version=1.0"
```

19.4.5 Use offset for pagination

```
# First page  
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
  "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?limit=1000"  
  
# Second page  
curl -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
  
↪ "https://api.dnsdb.info/dnsdb/v2/lookup/rrset/name/example.com?limit=1000&offset=1000"
```

Warning: Offset Limitations

The offset parameter is not a reliable way to walk all results over multiple queries because DNSDB recalculates results for each query and the order might not be preserved. Some results might be missing and some might be duplicated.

20 DNSDB Flex Search API Guide

20.1 Introduction

This page documents the **Flex Search API** extensions to DNSDB **APIv2**. **DNSDB APIv2** has two components: an enhanced Standard Search capability and Flexible Search. Flexible search adds ways to search DNSDB by regular expressions and globs (aka wildcarding). The [Farsight Flexible Search Reference Guide](#) should be read first, before reading this API programmer's guide.

In conjunction with the [DNSDB APIv2](#) page, this page describes how to make DNSDB Flexible search queries via the RESTful API.

20.2 Audience

This document is intended for programmers who want to write applications that can interact with the **Flex API** flexible search extensions to the RESTful DNSDB APIv2 using JSON and HTTP.

20.3 High level features

Flexible Search and its Flex API:

- Flex provides a flavor of “regular expressions” known as “extended regular expressions,” as used by the Unix `egrep` command. It supports most `egrep`-style regular expressions (except for capturing groups and backreferences), including simple string searches and partial label search. The particulars are known as “Farsight Compatible Regular Expressions” (FCRE) and are documented at [FCRE Reference Guide](#).
- Flex provides advanced wildcard search features, known in the Unix world as globs. They are documented at [Glob Reference Guide](#).
- Flex returns JSONL (newline delimited JSON) data, similar to what Standard Search returns, but the data are not the IETF Passive DNS - Common Output Format (COF) format.

20.3.1 URL encoding

When you're building URLs, some characters can't be used as normally written. These characters need to be written in their percent encoded form instead. Those special characters used in Flex regular expressions and globs patterns are:

special character	use in URLs
!	%21
”	%22
/	%2F
?	%3F
%	%25
[%5B
\	%5C

special character	use in URLs
]	%5D

See the *Example URLs* section below for some examples.

20.3.2 curl issue

In this document we show many *curl* examples. When using the Flex API with *curl* and you have use square brackets or curly brackets in the URL, you are *supposed* to encode them as mentioned above. *curl* usually works anyway without them encoded, but by default *curl* will interpret them itself. For example, if you ran a *curl* command with URL

```
.../dnsdb/v2/regex/rnames/example[1-2]
```

it would actually fetch two separate URLs:

```
.../dnsdb/v2/regex/rnames/example1
```

```
.../dnsdb/v2/regex/rnames/example2
```

to stop this, ideally use URL encoding; second best, use the `--globoff` (or `-g`) parameter to *curl*. All our *curl* examples show this parameter in use, even when not strictly necessary.

20.4 Summary of differences between APIv2 and the Flex API

- APIv1 and APIv2 only allow whole-label left-hand-side and right-hand-side wildcards, like `*.fsi.io` or `www.fsi.*`. Flexible search allows searches using regular expressions and globs (aka wildcarding) on `rnames` and `rdata`. - DNSDB calls searches on the owner name (i.e. left-hand-side of a DNS query-response) an “rrset” query. Flexible search calls that an “rnames” query, which is more consistent with DNS standards and reflects that the results from the search contain `rnames`, not full RRsets. - The URL structure below `/dnsdb/v2/` is completely different.
- The returned data stream, while still encapsulated in Farsight’s [Streaming API Framing protocol](#), are different JSON objects. - The Flex API does not provide `count`, `time_first`, and `time_last` in returned results.
- In the Flex API, the `aggr`, `humantime`, and `max_count` query parameters are *not* supported; the other APIv2 parameters are supported. - The `limit` parameter behaves slightly differently. In the Flex API, `limit` applies to the number of unique RRnames returned and not the number of JSON response objects returned, as it does in APIv2. That is, in the Flex API, `limit=1` might return 1 unique RRname, but multiple JSON response objects, one per RRType for that name.
- There are new error messages within the SAF “msg” object. - The X-RateLimit- headers are not returned in Flex API responses. Use the Standard Search API `rate_limit` request to get your quota information.

20.4.1 Summary of commonalities between APIv2 and the Flex API

- Both are encapsulated using the Farsight’s Streaming API Framing protocol.
- Both use the same API keys.
- Both share the same quotas.
- Both use the same HTTP Content-Types.

- Both return rrtype.
- Both return the same HTTP Response codes.

20.5 Streaming API Framing

The Flex API returns streamed data encapsulated in Farsight's [Streaming API Framing protocol](#). The SAF protocol is only applicable for HTTP responses that had an HTTP status of 200.

Here is an example Flex API query and response. Line breaks were added for readability, but the actual results were on one line:

```
https://api.dnsdb.info/dnsdb/v2/regex/rrnames/example\..*/ANY?limit=1
```

```
{"cond": "begin"}
{"obj":{"rrname":"example.s.ac.,"rrtype":"A"}}
{"cond": "succeeded"}
```

20.6 Authentication

The Flex API, as part of the DNSDB API 2.0, is provided over an encrypted HTTPS transport over the Internet at <https://api.dnsdb.info/>.

Authentication is performed by providing an API key (a long string of hexadecimal digits or dashes) in a special HTTP request header, **X-API-Key**. For example, if your API key is **d41d....8427e**, then the following HTTP header should be added to the HTTP request:

```
X-API-Key: d41d....8427e
```

If the **X-API-Key** header is not present, or the provided API key is not valid, a **403 Forbidden response** will be returned to the HTTP client.

20.7 Getting an API Key

To request a demonstration or a trial API key please contact the [DomainTools sales team](#).

20.8 Query URL structure

The query URL structure is:

```
api.dnsdb.info/dnsdb/v2/SEARCH_METHOD/WHAT_TO_
SEARCH/VALUE/RRTYPE?query-parameters
```

RRTYPE is optional, so the following is also valid:

```
api.dnsdb.info/dnsdb/v2/SEARCH_METHOD/WHAT_TO_SEARCH/VALUE?query-parameters
```

- /dnsdb/v2 - The first two path components.
- SEARCH_METHOD = Third-level path component:
 - regex - regular expression search
 - glob - full wildcarding
- WHAT_TO_SEARCH = Fourth-level path component:
 - rnames - search in rnames, supports “forward” searches based on the owner name of an RRset.

- rdata - search in rdata, supports “inverse” searches based on RData record values
- VALUE = Fifth-level path component:
 - For rnames queries, “LHS” DNS owner name regular expression or glob
 - For rdata queries, “RHS” normalized resource record value regular expression or glob
- RRTYPE = Sixth-level path component - Optional:
 - The official DNS RRType string value or TYPE# where # is the numeric value of the RRType.
 - * If an undefined RRType (but not TYPE# form) is specified (i.e. not in the DNS RFCs), then an HTTP 400 (Bad Request) error will be returned with message “Error: RRTYPE has an unsupported value”.
 - The RRtype ANY is modified somewhat from its usual meaning. A search for RRtype ANY will match any supported RRtype.
 - * ANY is the default if RRTYPE is not specified.
 - DNSSEC-related RRtypes are not permitted: CDS, CDNSKEY, DNSKEY, DS, DLV, NSEC, NSEC3, NSEC3PARAM, RRSIG, TA, nor our own ANY-DNSSEC.
 - * If one of those DNSSEC RRTypes is specified, then an HTTP 400 (Bad Request) error will be returned with message “Error: RRTYPE has an unsupported DNSSEC value”.
 - Only SOA, CNAME, HINFO, MX, NS, NAPTR, PTR, RP, SRV, SPF, or TXT (plus their TYPE# equivalents) are indexed for rdata queries. If another RRType is provided, then an HTTP 400 (Bad Request) error will be returned with message “Error: rdata searches are not allowed for the specified RRType”.
- If any of the path component values are not recognized, then an HTTP 400 (Bad Request) error will be returned with message “<X> has an unsupported value”.
- The maximum length of the VALUE path component and the exclude parameter value (which is described below) are 4096 characters each before URL encoding.

20.9 Optional query parameters

20.9.1 Time-fencing query parameters

The DNSDB records when a specific DNS record was first and last observed. You may filter results by time using the “time_first_before,” “time_first_after,” “time_last_before,” and “time_last_after” query parameters. These parameters expect a integer (Unix/Epoch time) with seconds granularity or a relative time in negative seconds, as relative times must be in the past from *now*.

Note that the DNSDB API server may have API key-dependent time-fencing restrictions that cannot be expanded with these parameters but can reduce the results (narrow the time ranges, or completely override them).

The following time-fencing optional parameters may be present in a query URL:

Parameter	Description
time_first_before	provide results before the defined timestamp for when the DNS record was first observed. For example, the URL parameter "time_first_before=1420070400" will only provide matching DNS records that were first observed before (or older than) January 1, 2015.
time_first_after	provide results after the defined timestamp for when the DNS record was first observed. For example, the URL parameter "time_first_after=-31536000" will only provide results that were first observed within the last year. (365 days * 24 hours/day * 60 minutes/hr * 60 minutes/sec = 31536000)
time_last_before	provide results before the defined timestamp for when the DNS record was last observed. For example, the URL parameter "time_last_before=1356998400" will only provide results for DNS records that were last observed before 2013.
time_last_after	provide results after the defined timestamp for when the DNS record was last observed. For example, the URL parameter "time_last_after=-2678400" will only provide results that were last observed after 31 days ago.

Combinations of the time parameters may be used to strictly provide or exclude results for specific time-ranges. For example, to only have results when the first observed date and the last observed date are both only in 2015, you can use "time_first_after=1420070399" combined with "time_last_before=1451606400". As another time combination example, to get DNS records that were first observed before 2012 and last observed within the last month (recently-observed records which have not changed in a very long time), use "time_first_before=1325376000" and relative "time_last_after=-2678400".

20.9.1.1 Other query parameters

The following other optional parameters may be present in a query URL:

Parameter	Description
exclude	The "exclude" parameter is used to exclude (i.e. filter-out) results that match it. It is described below this table.

Parameter	Description
limit	Limit for the number of unique rnames or rdata value results returned via these search methods. The default limit is set at 10,000. This limit can be raised or lowered by setting the "limit" query parameter. There is also a maximum number of results allowed; requesting a limit greater than the maximum will only return the maximum. See results_max in a rate_limit response for that maximum. If "?limit=0" is used then DNSDB will return the maximum number of results allowed. Obviously, if there are less results for the query than the requested limit, only the actual amount can be returned. Example: appending "?limit=20000" to the URL path will set the response limit to 20,000 results.
offset	How many rows to offset (e.g. skip) in the results. This implements an incremental result transfer feature, allowing you to view more of the available results for a single query. The rows are offset <i>prior</i> to the <i>limit</i> parameter being applied, therefore <i>offset</i> allows seeing additional results past a <i>limit</i> that matches the maximum number of results. Note that DNSDB recalculates the results for each query and the order of results might <i>not</i> be preserved. Therefore, this capability is <i>not</i> a valid way to walk all results over multiple queries -- some results might be missing and some might be duplicated. The actual offset that can be used is limited or for certain API keys, <i>offset</i> is not allowed -- see the Farsight DNSDB API Version 2 section on offset_max . The <i>offset</i> value must be a positive integer. The default is 0, which means do not offset the rows. Example to return the second million results (assuming results_max is 100000): "?limit=100000&offset=100000".
id	Client software specific identity of the user of the API client. Comprised of an alphanumeric string, a colon, and an alphanumeric string, limited to thirty characters. This may be logged by the DNSDB API server. There is no default. Example: "?id=dnsq:91e6245ad31387".
swclient	Name of the API client software generating the DNSDB query. Limited to twenty alphanumeric characters. This may be logged by the DNSDB API server. Farsight support can help you debug a new API client using this and the following parameter. There is no default. Example: "?swclient=dnsdbflex".

Parameter	Description
version	Version number of the API client software generating the query. Limited to twenty alphanumeric characters plus dash, underscore, and period. This may be logged by the DNSDB API server. There is no default. Example: “?version=1.1a”.

20.9.1.2 The *exclude* parameter

As mentioned above, the `exclude` parameter is used to exclude (i.e. filter-out) results that match it. Conceptually, this is like the shell pipeline: `egrep $VALUE <DNSDB | egrep -v $exclude`. Its value is a regular expression or glob, depending upon the `search_method`.

Here is an example with two `rrnames` searches, first without `exclude` and then excluding `rrnames` that end in `.com.`, `.site.`, `.bid.`, `.net.` or `.io`.

```
$ curl -g -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
'https://api.dnsdb.info/dnsdb/v2/regex/rrnames/^fsi.io/A?limit=3'
{"cond":"begin"}
{"obj":{"rrname": "fsi.io.", "rrtype": "A"}}
{"obj":{"rrname": "fsi.ioc.abc.cimpress.io.", "rrtype": "A"}}
{"obj":{"rrname": "fsi.io.ua.", "rrtype": "A"}}
{"cond":"succeeded"}
```

```
$ curl -g -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
'https://api.dnsdb.info/dnsdb/v2/regex/rrnames/^fsi.io/A?exclude=(.com|.site|bid|net|.io).$&limit=3'
{"cond":"begin"}
{"obj":{"rrname": "fsi.io.ua.", "rrtype": "A"}}
{"cond":"succeeded"}
```

20.9.1.3 Example URLs

Here are some example URLs showing a variety of URL components and parameters. We show each URL in its internal form and the form sent on the wire with URL encoding:

```
https://api.dnsdb.info/dnsdb/v2/regex/rrnames/example\..*/ANY?limit=1
```

```
https://api.dnsdb.info/dnsdb/v2/regex/rrnames/example%5C..%2A/ANY?limit=1
```

```
https://api.dnsdb.info/dnsdb/v2/glob/rrnames/example.*
```

```
https://api.dnsdb.info/dnsdb/v2/glob/rrnames/example.%2A
```

```
https://api.dnsdb.info/dnsdb/v2/glob/rdata/example.*TXT?limit=1\
&time_last_after=-2678400&exclude=*.com.&swclient=dnsdbflex
```

```
https://api.dnsdb.info/dnsdb/v2/glob/rdata/example.%2A/TXT?limit=1\
&time_last_after=-2678400&exclude=%2A.com.&swclient=dnsdbflex
```

20.9.2 Result formats

As an extension to DNSDB APIv2, the Flex API only supports one underlying result format, the “jsonl” format. “jsonl” has multiple equivalent names for it. It should be specified in an HTTP ACCEPT header as **application/x-ndjson**. See the HTTP Content-Types section below.

Pedantic note: all of our JSON / json outputs are really JSONL (newline delimited JSON). We use the terms JSON or json throughout this document unless that would cause confusion.

If there are no results to the query, then the HTTP status will be 200 with the following SAF objects returned:

```
{"cond":"begin"}
{"cond":"succeeded","msg":"No results found"}
```

This is different than the DNSDB APIv1.

If there are results to the query, each result object is an associative array with the following keys for the *terse* mode (which is the only currently supported mode). See the [Technical Reference Guide](#) *representation of rnames and rdata* section for details.

20.9.2.1 rnames results

Key	Description
rname	The owner name of the RRset in DNS presentation format.
rrtype	The resource record type of the RRset, either using the standard DNS type mnemonic, or an RFC 3597 generic type, i.e. the string TYPE immediately followed by the decimal RRtype number.

20.9.2.2 rdata results

Key	Description
rdata	The record data value.
rrtype	The resource record type of the resource record, either using the standard DNS type mnemonic, or an RFC 3597 generic type, i.e. the string TYPE immediately followed by the decimal RRtype number.
raw_rdata	The record data value as pairs of hex digits specifying a raw octet string. This value is used for pivoting from flexible search into standard search to get more details on rdata.

20.10 HTTP Content-Types

The preferred HTTP Content-Type is **application/x-ndjson**. Multiple synonyms for that are supported, even though the actual data returned is identical for all of those types. The API will return a Content-Type with that type.

The following are the allowed content-types in the HTTP **Accept:** header:

- **application/x-ndjson**
- **application/ldjson**
- **application/x-ldjson**
- **application/ndjson**
- **application/jsonl**
- **application/x-jsonl**

Additionally,

- ***/*** is a wildcard. If the Accept header specifies the ***/*** wildcard then the API will return **application/x-ndjson**.

If the request specifies multiple types in its **Accept:** header, then the first one that matches this supported list will be returned. The *;q= (q-factor weighting)* feature is not supported.

If none of the content-types in the **Accept:** header matches this supported list, then the API will return a 415 Unsupported Media Type error with content-type **text/plain**.

For example:

- If a request has “**Accept: application/jsonl**” it will get back “**Content-Type: application/jsonl**”.
- If a request has “**Accept: text/plain, application/x-ndjson**” it will get back “**Content-Type: application/x-ndjson**”.
- If a request has “**Accept: text/plain**” it will get back an HTTP 415 error.

For a 500 Internal Server Error, it returns **text/html** (no matter what was requested).

In case of other error responses generated by the DNSDB API, it returns **text/plain** (no matter what was requested).

For errors generated by nginx, such as a 404 Not Found for an unknown request URL, nginx returns **text/html**.

20.11 HTTP Response codes

The following is a table of HTTP response codes sent by the DNSDB Flex API. A single response code may be used with more than one message.

Response code	Description
200 OK	The request was successfully received, but errors may still be detected as it's further processed. Look at the final “cond” for success or failure of the request. See below for more information.
400 Bad Request	The URL is formatted incorrectly, such as unrecognized lower-level components. See below for more information.
401 Unauthorized	Your API key may not query this API version. Or the API key is not authorized (usually indicates the block quota is expired).

Response code	Description
403 Forbidden	The X-API-Key header is not present, the provided API key is not valid, or the Client IP address not authorized for this API key.
404 Not Found	The URL path had unrecognized or missing top-level components. This may also be returned for an unknown SEARCH_METHOD.
415 Unsupported Media Type	The Accept: header does not specify a supported content type for this query.
416 Requested Range Not Satisfiable	The offset value is greater than the maximum allowed or an offset value was provided when not permitted.
429 Too Many Requests	You have exceeded your quota and no new requests will be accepted at this time. For time-based quotas: The API key's daily quota limit is exceeded. The quota will automatically replenish, usually at the start of the next day. For block-based quotas: The block quota is exhausted. You may need to purchase a larger quota. For burst rate secondary quotas: There were too many queries within the burst window. The window will automatically reopen at its end. <i>Note: Burst rate quotas are not applicable currently for the Flex API.</i>
500 Internal Server Error	There is an internal error processing the request.
503 Service Unavailable	The limit of number of concurrent connections is exceeded.

20.11.1 Errors after a HTTP 200 (OK) status

Errors that occur after the query has started processing, and an HTTP 200 OK was returned, will be reported by a SAF "cond":"failed" result in its "msg" value. For example:

```
$ curl -g -k -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  'https://api-dev2.dnsdb.info/dnsdb/v2/regex/rnames/$^?limit=2'
...
< HTTP/1.1 200 OK
...
{"cond":"begin"}
{"cond":"failed","msg":"Regex syntax error: Invalid characters after '$' at 1"}
```

See the [Flexible Search Technical Reference Guide](#) section for descriptions and examples of these errors.

20.11.2 400 Bad Request errors

The response body after the HTTP 400 Bad Request will contain the following messages with a Content-Type: text/plain:

Message

Error: exception occurred– *If the URL is very mangled or an internal bug, please email *

Error: unable to parse request– *Query parameters have illegal values*

Error: SEARCH_METHOD has an unsupported value

Error: WHAT_TO_SEARCH has an unsupported value

Error: RRTYPE has an unsupported value

Error: RRTYPE has an unsupported DNSSEC value

Error: REQUEST_URI is missing path segment

Error: REQUEST_URI has too many path segments

Error: REQUEST_URI not available from environment– *An internal bug, please email *

Error: REQUEST_URI does not start with /dnsdb/v2/– *An internal bug, please email *

An example showing an illegal limit value (it should be a non-negative integer):

```
$ curl -g -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
  'https://api.dnsdb.info/dnsdb/v2/glob/rnames/*.google.*/ANY?limit=ILLEGAL' -
v
< HTTP/1.1 400 Bad Request
...
< Content-Type: text/plain
Error: unable to parse request
```

An example showing an unsupported WHAT_TO_SEARCH value:

```
$ dnsdbflex --regex 'test-search' -t badtype
dnsdbflex: warning: libcurl 400 [https://api.dnsdb.info/dnsdb/v2/regex/rnames/
  test-search/badtype?swclient=dnsdbflex&version=1.0.0]
dnsdbflex: warning: libcurl: [Error: RRTYPE has an unsupported value]
Query status: ERROR (Error: RRTYPE has an unsupported value)
```

dnsdbflex can be invoked with parameters that will trigger some of those 400 Bad Request error messages. For example:

```
$ dnsdbflex --regex &#039;test-search&#039;; -t badtype
dnsdbflex: warning: libcurl 400 [https://api.dnsdb.info/dnsdb/v2/regex/rnames/
  test-search/badtype?swclient=dnsdbflex&version=1.0.0]
dnsdbflex: warning: libcurl: [Error: RRTYPE has an unsupported value]
Query status: ERROR (Error: RRTYPE has an unsupported value)
```

20.12 Service limits and Quotas

The number of concurrent connections to a DNSDB API server may be limited. This limit is separate from the quota limit described below. If this limit is exceeded, the HTTP 503 “Service Unavailable” response code will be generated.

API keys have a primary quota, which limits the number of requests that can be made to the data-fetching endpoints. There are three types of quotas: time-based, block-based, and unlimited. The DNSDB API server tracks the usage of the quotas. If the quota’s limit is exceeded (if applicable), the HTTP 429 “Too Many Requests” response code will be generated

with message ‘Error: Rate limit exceeded’. If a block quota is expired, then a 401 “Unauthorized” response code will be generated with message ‘Error: Quota is expired’.

There may also be a secondary, burst rate, quota associated with an API key. The burst rate limits how many requests may be made in a short time window. For example, 5 requests in 360 seconds. The parameters for burst rate are burst size and burst window. *Note: Burst rate quotas are not applicable currently for the Flex API.*

You may query the `/dnsdb/v2/rate_limit` endpoint to obtain a JSON map containing a top-level map “rate” that contains quota information. See the [APIv2 Rate_limit response](#) section.

20.13 Example client

- `dnsdbflex`, a full-featured example API client written in C, is available from <https://github.com/farsightsec/dnsdbflex>
- Its [manual page](#).

20.14 Example results

For each example, the URL used to retrieve the results and sample curl commands with the current results (at the time of the writing of this document). For brevity, the results are limited to a total of two records in the examples below.

20.14.1 SAF format compared to dnsdbflex output format

For many examples, the equivalent `dnsdbflex` invocation and outputs are provided. Note that the json output from `dnsdbflex` is the “unwrapped” form of the SAF format.

For example, the following SAF:

```
{"cond":"begin"}
{"obj":{"rrname": "fsi.iosdjf.cn.", "rrtype": "CNAME"}}
{"obj":{"rrname": "fsi.iosdnb.cn.", "rrtype": "CNAME"}}
{"cond":"succeeded"}
```

is output from `dnsdbflex` when using its default json (-j) format as:

```
{"rrname": "fsi.iosdjf.cn.", "rrtype": "CNAME"}
{"rrname": "fsi.iosdnb.cn.", "rrtype": "CNAME"}
```

and is output from `dnsdbflex` in -F batch mode output format as:

```
rrset/name/fsi.iosdjf.cn./CNAME
rrset/name/fsi.iosdnb.cn./CNAME
```

20.14.2 Setting DNSDB_API_KEY

The examples assume the shell variable `DNSDB_API_KEY` is set to your API key:

```
DNSDB_API_KEY=0123456890abcd...
```

20.14.2.1 1. Search for all rrnames whose owner name matches regular expression `^fsi\.io`, limited to 2 rrnames. Note: This is *not* right anchored

Pattern: `^fsi\.io`

URL

```
/dnsdb/v2/regex/rrnames/^fsi\.io/ANY?limit=2
```

Sample query

```
$ curl -g -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
'https://api.dnsdb.info/dnsdb/v2/regex/rrnames/^fsi\.io/ANY?limit=2'
```

Response:

```
{"cond":"begin"}
{"obj":{"rrname": "fsi.iosdjf.cn.", "rrtype": "CNAME"}}
{"obj":{"rrname": "fsi.iosdnb.cn.", "rrtype": "CNAME"}}
{"cond":"succeeded"}
```

20.14.3 2. Search for all rrnames whose owner name matches regular expression `^fsi\.io$`, limited to 2 rrnames. Note: This *is* right anchored

This shows a problem where we forgot to add a regex expression that matches a dot before the right anchor.

Pattern: `^fsi\.io$` (missing trailing dot)

URL

```
/dnsdb/v2/regex/rrnames/^fsi\.io$/ANY?limit=2
```

Sample query

```
$ curl -g -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
'https://api.dnsdb.info/dnsdb/v2/regex/rrnames/^fsi\.io$/ANY?limit=2'
```

Response:

```
{"cond":"begin"}
{"cond":"succeeded","msg":"No results found"}
```

To **fix** this query, try one of the following patterns:

```
/dnsdb/v2/regex/rrnames/^fsi\.io\.$/ANY?limit=2
/dnsdb/v2/regex/rrnames/^fsi\.io\.$/ANY?limit=2
```

20.14.4 3. Search for all rdata which matches regular expression `^fsi\.io`, limited to 2 results

Pattern: `^fsi\.io`

URL

```
/dnsdb/v2/regex/rdata/^fsi\.io/ANY?limit=2
```

Sample query

```
$ curl -g -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \
'https://api.dnsdb.info/dnsdb/v2/regex/rdata/^fsi\.io/ANY?limit=2'
```

Response:

```
{"cond":"begin"}
{}
{"obj":{"rdata": "fsi.io. hostmaster.fsi.io.", "rrtype": "SOA",
"raw_rdata": "0366736902696F000A686F73746D61737465720366736902696F00"}}
```

```
{"obj":{"rdata": "fsi.ioza15pw68tk.k0l33r.info.", "rrtype": "CNAME",
  "raw_rdata": "036673690C696F7A61313570773638746B066B306C33337204696E666F00"}}
{"cond":"succeeded"}
```

Note: The output includes one keep alive message (the empty {} object).

dnsdbflex equivalent

```
$ dnsdbflex --regex '^fsi\.io' -l 2 -s rdata
```

dnsdbflex output:

```
{"rdata":"fsi.io. hostmaster.fsi.io.", "rrtype":"SOA"
  "raw_rdata":"0366736902696F000A686F73746D61737465720366736902696F00"}
{"rdata": "fsi.ioza15pw68tk.k0l33r.info.", "rrtype": "CNAME",
  "raw_rdata": "036673690C696F7A61313570773638746B066B306C33337204696E666F00"}
```

20.14.5 4. Use dnsdbflex to search all rnames matching a complex range glob

This shows how some of the special characters in the glob are quoted by dnsdbflex.

Glob pattern: `[!a-vx-z]sillyputty.info.`

Command:

```
$ dnsdbflex --glob '[!a-vx-z]sillyputty.info.'
```

URL issued:

```
/dnsdb/v2/glob/rnames/a-vx-zsillyputty.info.?swclient=dnsdbflex&version=1.0.0&limit=1
```

API response:

```
{"cond":"begin"}
{"rrname": "wsillyputty.info.", "rrtype": "A"}
{"rrname": "wsillyputty.info.", "rrtype": "NS"}
{"rrname": "wsillyputty.info.", "rrtype": "SOA"}
{"rrname": "wsillyputty.info.", "rrtype": "MX"}
{"rrname": "wsillyputty.info.", "rrtype": "TXT"}
{"rrname": "wsillyputty.info.", "rrtype": "AAAA"}
{"cond":"succeeded"}
```

dnsdbflex output:

```
{"rrname": "wsillyputty.info.", "rrtype": "A"}
{"rrname": "wsillyputty.info.", "rrtype": "NS"}
{"rrname": "wsillyputty.info.", "rrtype": "SOA"}
{"rrname": "wsillyputty.info.", "rrtype": "MX"}
{"rrname": "wsillyputty.info.", "rrtype": "TXT"}
{"rrname": "wsillyputty.info.", "rrtype": "AAAA"}
```

Note: Even though `limit=1` was specified, we got back 6 results because there is only one unique rname in those results (with different RRtypes).

20.14.6 5. Request with missing lower-level components

This call shows a request with a missing lower-level component (the `/rnames` component is missing):

Sample query

```
$ curl -g -H "Accept: application/x-ndjson" -H "X-API-Key: $DNSDB_API_KEY" \  
  'https://api.dnsdb.info/dnsdb/v2/regex/^fsi\.io/ANY?limit=2' -v
```

Response: HTTP 400 Bad Request with body:

Error: WHAT_T0_SEARCH has an unsupported value

21 DNSDBQ (Query Tool) Guide

21.1 Introduction

```
dnsdbq [-acdfgGhIjmQsUv468] [-A timestamp] [-B timestamp] [-b bailiwick] [-i ip]
↪ [-J input_file] [-k sort_keys] [-L output_limit] [-l query_limit] [-M
↪ max_count] [-N raw_name] [-n name] [-O offset] [-p output_type] [-R raw_rrset]
↪ [-r rrset] [-T transform[f,...]] [-t rrtype] [-u server_sys] [-V verb]
```

21.2 Description

dnsdbq constructs and issues queries to Passive DNS systems which return data in the IETF Passive DNS Common Output Format. It is commonly used as a production command line interface to such systems.

dnsdbq displays responses in various formats. Its default query type is a “lookup” query. As an option, it can issue a “summarize” query type. Different Passive DNS systems or versions of those systems may implement different query features.

21.3 Farsight DNSDB

Farsight Security’s *DNSDB* is one such Passive DNS system. DNSDB implements both APIv1 and APIv2 interfaces. APIv1 is accessed by specifying system *dnsdb1*. APIv2 is accessed by specifying system *dnsdb2*.

You’ll need to get an API key from Farsight to use **dnsdbq** with DNSDB.

Farsight’s passive DNS infrastructure performs a complex process of “bailiwick reconstruction” where an RRset’s position within the DNS hierarchy is approximated. This serves two purposes:

1. Provide context of the location of a given DNS record within the DNS hierarchy
2. Prevent “untrustworthy” records that are a result of intentional or unintentional cache poisoning attempts from being replicated by downstream consumers.

For example, given the fully qualified domain name **www.dachshund.example.com**, valid bailiwicks would be **dachshund.example.com**, **example.com**, or **com**.

21.4 Options

-a: enables ASINFO/CIDR annotation for IP addresses in A (IPv4 address) RRsets. Depending on the data source given by **-D**, the result might be of a least-specific, most-specific, or intermediate route.

-A timestamp: Specify a backward time fence. Only results seen by the passive DNS on or after this time will be selected. See also **-c**. See the **TIMESTAMP FORMATS** section for more information about this.

-B timestamp Specify a forward time fence. Only results seen by the passive DNS sensor network on or before this time will be selected. See also **-c**. See the **TIMESTAMP FORMATS** section for more information about this.

-b *bailiwick*: specify bailiwick (only valid with **-r** queries).

-c: by default, **-A** and **-B** (separately or together) will select partial overlaps of database tuples and time search criteria. To match only complete overlaps, add the **-c** (“completeness”) command line option (this is also known as “strict” mode).

-D: specify the data source for ASINFO/CIDR annotations, if enabled by **-a**. Default is `asn.routeviews.org`, but you may wish to try `aspath.routeviews.org`.

-d: enable debug mode. Repeat for more debug output.

-f: specify batch lookup mode allowing one or more queries to be performed. Queries will be read from standard input and are expected to be in one of the following formats:

- RRset (raw) query: **rrset/name/NAME[/RRTYPE[/BAILIWICK]]**
- RRset (raw) query: **rrset/raw/HEX[/RRTYPE[/BAILIWICK]]**
- Rdata (name) query: **rdata/name/NAME[/RRTYPE]**
- Rdata (IP address) query: **rdata/ip/ADDR[,PFXLEN]**
- Rdata (raw) query: **rdata/raw/HEX[/RRTYPE]**
- Change query options: **\$OPTIONS {options}**

Where **options** ::=

- **-A** timestamp
- **-B** timestamp
- **-c**
- **-g**
- **-G**
- **-l** query_limit
- **-L** output_limit
- **-O** offset

\$OPTIONS alone on a line allows command line options to be changed mid-batch. If no options are given, the query parameters will be reset to those given on the command line, if any, or else to defaults.

A line starting with a **#** will be ignored as a comment.

Any internal slash (/) or comma (,) characters within the search names of a batch entry must be URL-encoded (for example, or).

For raw queries, the HEX value is an even number of hexadecimal digits specifying a raw octet string. The “raw” wire-format encodings are standardized. The embedding of these in `dnstable` is documented in the *dnstable-encoding(5)* manual page.

In batch lookup mode, each answer will be followed by a **--** marker, so that programmatic users will know when it is safe to send the next lookup, or if lookups are pipelined, to know when one answer has ended and another begun. This option cannot be mixed with **-n**, **-r**, **-R**, or **-i**. See the EXAMPLES section for more information on how to use **-f**.

If two **-f** options are given, then each answer will also be preceded by a **++** marker giving the query string (as read from the batch input) in order to identify each answer when a very large batch input is given, and the **--** marker will include an error/noerror indicator and a short message describing the outcome. With two **-f** options and also **-m**, answers can appear in a different order than the batched questions.

The **++** and **--** markers are not valid JSON, CSV, or DNS (text) format, so caution is required. (See **-m** option below.)

-g: return graveled results. Default is to return aggregated results (rocks, vs. gravel). Gravel is a feature for providing Volume Across Time.

-G: undo the effect of **-g**, this returning rocks rather than gravel. (Used in \$OPTIONS in batch files.)

-h: emit usage and quit.

-I: request information from the API server concerning the API key itself, which may include rate limit, query quota, query allowance, or privilege levels; the output format and content is dependent on the server_sys argument (see **-u**) and upon the **-p** argument. **-I -p json** prints the raw info. **-I -p text** prints the information in a more understandable textual form, including converting any epoch integer times into UTC formatted times.

-i ip: specify rdata ip (“right-hand side”) query. The value is one of an IPv4 address, an IPv6 address, an IPv4 network with prefix length, an IPv4 address range, or an IPv6 network with prefix length. If a network lookup is being performed, the delimiter between network address and prefix length is a single comma (“,”) character rather than the usual slash (“/”) character to avoid clashing with the HTTP URI path name separator. See EXAMPLES section for more information about separator substitution rules.

-J input_file: opens input_file and reads newline-separated JSON objects therefrom, in preference to **-f** (batch mode) or query mode. This can be used to reprocess the output from a prior invocation which used **-j** (-p json). Sorting, limits, and time fences will work. Specification of a domain name, RRtype, Rdata, or offset is not supported at this time. If input_file is “-” then standard input (stdin) will be read.

-j: specify newline delimited json output mode.

-k sort_keys: when sorting with **-s** or **-S**, selects one or more comma separated sort keys, among “first”, “last”, “duration”, “count”, “name”, and/or “data”. The default order is be “first,last,duration,count,name,data” (if sorting is requested.) Names are sorted right to left (by TLD then 2LD etc). Data is sorted either by name if present, or else by numeric value (e.g., for A and AAAA RRsets.) Several **-k** options can be given after different **-s** and **-S** options, to sort in ascending order for some keys, descending for others.

-l query_limit: query for that limit’s number of responses. If specified as 0 then the DNSDB API server will return the maximum limit of results allowed. If **-l**, is not specified, then the query will not specify a limit, and the DNSDB API server may use its default limit.

-L output_limit: clamps the number of objects per response (under **-[R|r|N|n|i|f]**) or for all responses (under **-[fm|ff|ffm]**) output to **-output_limit**. If unset, and if batch and merge modes have not been selected with the **-f** and **-m** options, then the **-L** output limit defaults to the **-l** limit’s value. Otherwise the default is no output limit.

-M max_count: for the summarize verb, stops summarizing when the count reaches that max_count, which must be a positive integer. The resulting total count may exceed max_count as it will include the entire count from the last rset examined. The default is to not constrain the maximum count. The number of rsets summarized is also limited by the query_limit.

-m: used only with **-f**, this causes multiple (up to ten) API queries to execute in parallel. In this mode there will be no “-” marker, and the combined output of all queries is what will be subject to sorting, if any. If two **-f** flags are specified with **-m-**, the output will not be merged, can appear in any order, will be sorted separately for each response, and will have normal ‘-’ / ‘++’ markers. (See **-f** option above.)

-N HEX: specify raw **-rdata** data (“right-hand side”) query. HEX is as described above.

-n name: specify **-rdata** name (“right-hand side”) query. The value is a DNS domain name in presentation format, or a left-hand (“.example.com”) or right-hand (“www.example.”) wildcard

domain name. Note that left-hand wildcard queries are somewhat more expensive than right-hand wildcard queries.

-O *offset*: to offset by *#offset* the results returned by the query. This gives you incremental results transfers. Cannot be negative. The default is 0.

-p *output_type*: select output type. Specify:

- **text** for presentation output meant to be human-readable. This is the default.

dns is a synonym, for compatibility with older programmatic callers.

- **json** for newline delimited JSON output.
- **csv** for comma separated value output. This format is information losing, since it cannot express multiple resource records that are in a single RRset. Instead, each resource record is expressed in a separate line of output. See the

DNSDBQ_TIME_FORMAT environment variable below for controlling how timestamps are formatted for this option.

-q: makes the program reticent about warnings.

-R *HEX*: specify raw **rrset** owner data (“left-hand side”) query. HEX is as described above.

-r *rrset*: specify rrset (“left-hand side”) name query.

-s: sort output in ascending key order. Limits (if any) specified by **I** and **L** will be applied before and after sorting, respectively. In batch mode, the **-f**, **-ff**, and **-ffm** option sets will cause each batch entry’s result to be sorted independently, whereas with **-fm**, all outputs will be combined before sorting. This means with **-fm** there will be no output until after the last batch entry has been processed, due to store and forward by the sort process.

-S: sort output in descending key order. See discussion for **-s** above.

-T *transform[,...]*: specify one or more transforms to be applied to the output:

- **datefix**: always show dates in human readable format (so, not in database format). This will be the format selected by the **DNSDBQ_TIME_FORMAT** environment variable, if set.
- **reverse**: show the DNS owner name (rrname) in TLD-first order (so, COM.EXAMPLE rather than EXAMPLE.COM).
- **chomp**: strip away the trailing dot (.) from the DNS owner name (rrname).

-t *rrtype*: specify the resource record type desired. Default is ANY. If present, this option should precede any **-R**, **-r**, **-N**, or **-n** options. This option is not allowed if the **-i** option is present. Valid values include those defined in DNS RFCs, including ANY. A special-case supported in DNSDB is ANY-DNSSEC, which matches on CDS, CDNSKEY, DNSKEY, DS, DLV, NSEC, NSEC3, NSEC3PARAM, RRSIG, and TA resource record types.

-u *server_sys*: specifies the syntax of the RESTful URL, default is “dnsdb”.

-V *verb*: The verb to perform, i.e. the type of query, either “lookup” or “summarize”. The default is the “lookup” verb. As an option, you can specify the “summarize” verb, which gives you an estimate of result size. At-a-glance, it provides information on when a given domain name, IP address or other DNS asset was first-seen and last-seen by the global sensor network, as well as the total observation count.

-U: turns off TLS certificate verification (unsafe).

-v: report the version of dnsdbq and exit.

-4: use to force connecting to the DNSDB server via IPv4.

-6: use to force connecting to the DNSDB server via IPv6.

-8: Normally dnssdbq requires that **-n** or **-r** arguments are 7-bit ASCII clean. Non-ASCII values should be queried using PUNYCODE IDN encoding. This **-8** option allows using arbitrary 8 bit values.

21.5 Timestamp Formats

Timestamps may be one of following forms.

- positive unsigned integer : in Unix epoch format.
- negative unsigned integer : negative offset in seconds from now.
- YYYY-MM-DD [HH:MM:SS] : in absolute form, in UTC time, as DNSDB does its fencing using UTC time.
- %uw%ud%uh%um%us : the relative form with explicit labels (*w*=weeks, *d*=days, *h*=hours, *m*=minutes, *s*=seconds). Calculates offset from UTC time, as DNSDB does its fencing using UTC time.

When using batch mode with the second or forth cases, using relative times to now, the value for “now” is set when dnssdbq starts.

A few examples of how to use timefencing options.

```
# tuples ending after Aug 22, 2015 (midnight)
$ dnssdbq ... -A 2015-08-22
# tuples starting before Jan 22, 2013 (midnight)
$ dnssdbq ... -B 2013-01-22
# tuples starting or ending from 2015 (midnight to midnight)
$ dnssdbq ... -B 2016-01-01 -A 2015-01-01
# tuples ending after 2015-08-22 14:36:10
$ dnssdbq ... -A "2015-08-22 14:36:10"
# tuples ending within the last 60 minutes
$ dnssdbq ... -A "-3600"
# tuples ending after "just now"
$ date +%s
1485284066
$ dnssdbq ... -A 1485284066
# batch mode with only tuples ending within last 60 minutes,
# even if feeding inputs to dnssdbq in batch mode takes hours.
$ dnssdbq -f ... -A "-3600"
```

21.6 Examples

A few examples of how to specify IP address information.

```
# specify a single IPv4 address
$ dnssdbq ... -i 128.223.32.35
# specify an IPv4 CIDR
$ dnssdbq ... -i 128.223.32.0/24
# specify a range of IPv4 addresses
$ dnssdbq ... -i 128.223.32.0-128.223.32.32
```

Perform an RRset query for a single A record for **farsightsecurity.com**. The output is serialized as JSON and is piped to the **jq** program (a command-line JSON processor) for pretty printing.

```
$ dnsdbq -r farsightsecurity.com/A -l 1 -j -a | jq .
{
  "count": 6350,
  "time_first": 1380123423,
  "time_last": 1427869045,
  "rrname": "farsightsecurity.com.",
  "rrtype": "A",
  "bailiwick": "farsightsecurity.com.",
  "rdata": [
    "66.160.140.81"
  ],
  "dnsdbq-rdata": [
    {
      "asinfo": [ 6939 ],
      "cidr": "66.160.128.0/18",
      "rdata": "66.160.140.81"
    }
  ]
}
```

Note the “dnsdbq-rdata” element added due to the use of the -a option.

Perform a batched operation for a several different **rrset** and **rdata** queries. Output is again serialized as JSON and redirected to a file.

```
$ cat batch.txt
rrset/name/\*.wikipedia.org
rrset/name/\*.dmoz.org
rrset/raw/0366736902696f00/A
rdata/name/\*.pbs.org
rdata/name/\*.opb.org
rdata/ip/198.35.26.96
rdata/ip/23.21.237.0,24
rdata/raw/0b763d73706631202d616c6c
$ dnsdbq -j -f < batch.txt > batch-output.json
$ head -1 batch-output.json | jq .
{
  "count": 2411,
  "zone_time_first": 1275401003,
  "zone_time_last": 1484841664,
  "rrname": "wikipedia.org.",
  "rrtype": "NS",
  "bailiwick": "org.",
  "rdata": [
    "ns0.wikimedia.org.",
    "ns1.wikimedia.org.",
    "ns2.wikimedia.org."
  ]
}
```

21.7 ASINFO/CIDR Lookups

When the **-a** option is used, every address seen in a response will cause a DNS lookup under the domain specified by the **-D** option. This stream of DNS queries might be an intolerable information leak depending on the nature of the underlying research, and it could also lead to unusably bad performance depending on the placement of your configured recursive DNS service.

For best results, always use an on-server or on-LAN recursive DNS service, and consider whether to configure that recursive DNS service to be a “stealth secondary” of the zone denoted by the **-D** option. For the default **-D** value, more information can be found online at <http://archive.routeviews.org/dnszones/>.

Use of DNS lookups to retrieve ASINFO/CIDR metadata can be extremely fast and surveillance-free, but some attention must be paid in order to obtain that outcome. For occasional low-volume use, your current recursive DNS placement and configuration is probably good enough.

Note that while Passive DNS information is historical, the ASINFO/CIDR annotations made possible using the **-a** and **-D** options are based on current information. Internet routing system information may have changed since the DNS data was recorded. More information about this can be found online at <https://github.com/dnsdb/dnsdbq/blob/master/README>.

21.8 Files

~/isc-dnsdb-query.conf **~/dnsdb-query.conf** **/etc/isc-dnsdb-query.conf** or **/etc/dnsdb-query.conf**: configuration file which can specify the API key, etc. variables. The first of these which is readable will be used, alone, in its entirety. See the **DNSDBQ_CONFIG_FILE** environment variable which can specify a different configuration file to use.

The variables which can be set in the configuration file are as follows:

- **DNSDBQ_SYSTEM**: contains the default value for the **-u** option described above.
- **DNSDB_API_KEY**, **APIKEY** contains the user’s DNSDB apikey (no default).
- **DNSDB_SERVER**: contains the URL of the DNSDB API server (default is), and optionally the URI prefix for the database. The default URI prefix for system “dnsdb2” is “/dnsdb/v2/lookup”; the default for “dnsdb1” is “/lookup”.
- **CIRCL_AUTH**, **CIRCL_SERVER**: enable access to a passive DNS system compatible with the CIRCL.LU system.

21.9 Environment

- **DNSDBQ_CONFIG_FILE**: specifies the configuration file to use, overriding the internal search list.
- **DNSDB_API_KEY**: contains the user’s apikey. The older **APIKEY** environment variable has been retired, though it can still be used in the configuration file.
- **DNSD_SERVER**: contains the URL of the DNSDB API server, and optionally a URI prefix to be used (default is “/lookup”). If not set, the configuration file is consulted.
- **DNSDBQ_TIME_FORMAT**: controls how human readable date times are displayed from the **-p csv** output format. If “iso” then ISO8601 (RFC3339) format is used, for example; “2018-09-06T22:48:00Z”. If “csv” then an Excel CSV compatible format is used; for example, “2018-09-06 22:48:00”.

21.10 Exit Status

Success (exit status zero) occurs if a connection could be established to the back end database server, even if no records matched the search criteria. Failure (exit status nonzero) occurs if no connection could be established, perhaps due to a network or service failure, or a configuration error such as specifying the wrong server hostname.

22 DNSDB Flexible Search Query Tool Reference

22.1 Synopsis

```
dnsdbflex [-cdFjhqTUv46] [--exclude glob|regular_expression] [--force] [--glob glob]
[--mode terse] [--regex regular_expression] [-A timestamp] [-B timestamp] [-l
query_limit] [-O offset] [-s search_what] [-t rtype] [-u server_sys]
```

22.2 DESCRIPTION

dnsdbflex constructs and issues flexible search queries to Farsight Security's DNSDB system. The flexible searches include regular expressions and globs (i.e. full wildcarding). The results from **dnsdbflex** can be displayed directly as JSON or emitted in the **dnsdbq** batch file input format. Using the batch file format allows "pivoting" from an flexible search into complete DNSDB results.

Values to **--glob**, **--regex**, or **--exclude** are called search expressions.

Search expressions must match the the data indexed in DNSDB's flexible search database. All DNS rnames end in a dot. All rdata indexed in the database will end in a dot if it's a host name or will end in a double quote for other data. A search expression that does not conform with that may be a wasted query. For example, **dnsdbflex --glob '*.fsi.io'** will not match anything because globs are right-anchored and that search expression does not end in a dot. In glob search expressions, **dnsdbflex** normally detects such violations and disallows them. Add a trailing dot to match something, **dnsdbflex --glob '*.fsi.io.'**

dnsdbflex doesn't detect such violations in regex search expressions. For example, **dnsdbflex --regex '*.fsi.io' **willnotmatchanythingbut* **dnsdbflex - regex'..fsi.io.*** does (note the dot before the dollar sign).

Search expressions must contain 7 bit clean, printable ASCII characters. Use Punycode IDN encoding to search for IDN domain names. Use \DDD (where DDD is the decimal value of the character) to match non-printable characters in rdata strings.

See the [Farsight Compatible Regular Expression Reference Guide](#) (FCRE) for a description of the wildcarding syntax, as well as more examples.

You'll need to get an API key from Farsight to use **dnsdbflex** with DNSDB. Note that certain types of API keys may not be allowed to use this API, in which case, **dnsdbflex** will fail with error message "The type of API key is not allowed to use the DNSDB Flex API".

22.3 OPTIONS

Either **--glob** or **--regex** must be specified. Both cannot be specified at the same time.

--exclude *glob|regular_expression*: Filters out results selected by a glob or regular expression. If **--glob** was specified, then **--exclude** takes a glob. If **--regex** was specified, then **--exclude** takes a regular expression.

--force: Issue search queries even if rejected by **dnsdbflex**'s pattern checks.

--glob *glob*: Specify that **dnsdbflex** should do a glob search. Only the * and [] glob operators are supported. Can abbreviate as **--g**.

--mode terse: Specify mode of information to return in results.

- **terse**: Can abbreviate as **t**. This is the only value currently supported and is the default, so the **--mode** option need not be specified.
- For rnames queries, returns the rname and rrtype.
- For rdata queries, returns normalized rdata, rrtype, and raw_rdata.

--regex regular_expression: Specify that **dnsdbflex** should do a regular expression search in the FCRE syntax. Can abbreviate as **--r**.

-A timestamp: Specify a backward time fence. Only results seen by the passive DNS sensor network on or after this time will be selected. See also **-c**. See the TIME FENCING section for more information.

-B timestamp: Specify a forward time fence. Only results seen by the passive DNS sensor network on or before this time will be selected. See also **-c**. See the TIME FENCING section for more information.

-c: By default, **-A** and **-B** (separately or together) will select partial overlaps of database tuples and time search criteria. To match only complete overlaps, add the **-c** (“completeness”) command line option (this is also known as “strict” mode). See the TIME FENCING section for more information.

-d: enable debug mode. Repeat for more debug output.

-F: specify batch output mode, outputting results in the batch format that **dnsdbq -f** can read.

-F includes the rrtype in the batch file format queries -- in contrast to **-T** described below.

- If searching for rdata, if an rdata value is not printable or contains whitespace, it will emit it using the raw_rdata hex value and output a comment line giving the non-raw format.
- See **dnsdbq(1)** for documentation of the batch format.

-h: emit usage and quit.

-j: output in JSON format, which is the default. **-j** and **-F** are mutually exclusive.

-l query_limit: query for that limit’s number of responses. If specified as 0 then the DNSDB API server will return the maximum limit of results allowed. If **-l** is not specified, then the query will not specify a limit, and the DNSDB API server may use its default limit.

-O offset: to offset by #offset the results returned by the query. This gives you approximate incremental results transfers. Results can be reordered between queries, so using progressively higher offsets is not guaranteed to return all results without duplicates or gaps. Offset cannot be negative and the default is 0.

-q: (quiet) suppresses sending warning messages to stderr.

-s rnames|rdata: what data to search.

- **rnames**: Search in the rnames part of DNS records, aka the left-hand side. Can abbreviate as **n**. This is the default.
- **rdata**: Search in the rdata part of DNS records, aka the right-hand side. Can abbreviate as **d**.

-t rrtype: specify the resource record type desired. Default is ANY.

- For rnames queries, valid rrtypes include those defined in DNS RFCs, including ANY, except DNSSEC types are not allowed (ANY-DNSSEC, CDNSKEY, CDS, DLV, DNSKEY, DS, NSEC, NSEC3, NSEC3PARAM, RRSIG, and TA resource record types). Also valid are TYPE# values.

- For rdata queries, only the following rrtypes are valid: CNAME, HINFO, MX, NAPTR, NS, PTR, RP, SOA, SPF, SRV, and TXT. Also valid are their TYPE# values.

-T: Like **-F** but does not include the rrtype in the batch file queries. This allows pivots to match against all available rrtypes. The batch output will also include a comment for each line including the rrtype.

-u server_sys: specifies the syntax of the RESTful URL. The only system currently supported is “dnsdb2”, which is the default.

-U: turns off TLS certificate verification (unsafe).

-v: report the version of **dnsdbflex** and exit.

-4: force connecting to the DNSDB server via IPv4.

-6: force connecting to the DNSDB server via IPv6.

22.4 EXAMPLES

```
# Regular expression search of all rnames that contain a coke label,
# for all rrtypes, limit of 10 results.
$ dnsdbflex --regex '.*.coke.*' -l 10

# Same query without using default values
$ dnsdbflex --regex '.*.coke.*' -l 10 -s rnames --mode terse

# Glob search of all names that contain a coke label and have an 'A' RRTYPE.
$ dnsdbflex --glob '.*.coke.*' -l 10 -t A

# Pivot those results into dnsdbq for full DNSDB API results in json
# form. Note that up to 11 DNSDB query quota units will be consumed,
# 1 by dnsdbflex and 10 by dnsdbq. If we did not specify the RRTYPE
# to dnsdbflex, then it might return more than 10 results (one for
# each RRTYPE for each name) and we'd use more than 11 DNSDB query
# quota units.
$ dnsdbflex --glob '.*.coke.*' -l 10 -t A -F | dnsdbq -f -j

# Get names containing "coke" but then exclude all those containing "diet".
$ dnsdbflex --glob '.*.coke.*' --exclude '.*diet.*' -l 10

# Same query, but using regular expressions
$ dnsdbflex --regex '.*.coke.*' --exclude '.*diet.*' -l 10
```

22.5 TIME FENCING

Farsight’s DNSDB flexible search provides time fencing options for searches. The **-A** and **-B** options take a timestamp as an argument. The timestamps may be one of following forms.

- positive unsigned integer : in Unix epoch format.
- negative unsigned integer : negative offset in seconds from now.
- YYYY-MM-DD [HH:MM:SS] : in absolute form, in UTC time, as DNSDB does its fencing using UTC time.
- %uw%ud%uh%um%us : the relative form with explicit labels (w=weeks, d=days, h=hours, m=minutes, s=seconds). Calculates offset from UTC time, as DNSDB does its fencing using UTC time.

A few examples of how to use time fencing options:

```
# Responses after Aug 22, 2015 (midnight),
# excluding records ALSO seen before that time.
$ dnssdbflex... -c -A 2015-08-22
```

```
# Responses from 2015 (midnight to midnight),
# but not excluding records ALSO seen outside that time range.
$ dnssdbflex... -B 2016-01-01 -A 2015-01-01
```

Certain settings for time fences may be used to accelerate queries for rnames and rdata values which have been recently observed or which were first observed in the distant past. Time fencing may accelerate the query if either **-A** or **-B** (but not both) are supplied without **-c**.

A few examples of how to use time fencing options where the query may be accelerated:

```
# Responses after 2015-08-22 14:36:10,
# but not excluding records ALSO seen before that time.
$ dnssdbflex... -A "2015-08-22 14:36:10"
```

```
# Responses from the last 60 minutes,
# but not excluding records ALSO seen before that time.
$ dnssdbflex... -A "-3600"
```

```
# Responses after Aug 22, 2015 (midnight),
# but not excluding records ALSO seen before that time.
$ dnssdbflex... -A 2015-08-22
```

```
# Responses before Jan 22, 2013 (midnight),
# but not excluding records ALSO seen after that time.
$ dnssdbflex... -B 2013-01-22
```

22.6 FILES

~/dnssdb-query.conf or **/etc/dnssdb-query.conf**: configuration file which can specify the API key, etc. variables. The first of these which is readable will be used, alone, in its entirety. See the **DNSDBQ_CONFIG_FILE** environment variable which can specify a different configuration file to use.

For backwards compability, **~/isc-dnssdb-query.conf** and **/etc/isc-dnssdb-query.conf** are also valid, but deprecated.

The variables which can be set in the configuration file are as follows:

- **DNSDB_API_KEY, APIKEY**: contains the user's API key (no default).
- **DNSDB_SERVER**: contains the URL of the DNSDB API server (default is <https://api.dnssdb.info/>), and optionally the URI prefix for the database.
- **DNSDBQ_SYSTEM**: contains the default value for the *u* option described above. Can only be "dnssdb2". If unset, **dnssdbflex** will probe for any configured system.

22.7 ENVIRONMENT

The following environment variables affect the execution of **dnssdbflex**:

- **DNSDBQ_CONFIG_FILE**: specifies the configuration file to use, overriding the internal search list.

- **DNSDB_API_KEY**: contains the user's apikey. The older APIKEY environment variable has been retired, though it can still be used in the configuration file.
- **DNSDB_SERVER** contains the URL of the DNSDB API server, and optionally a URI prefix to be used. If not set, the configuration file is consulted.
- **DNSDBQ_SYSTEM** See DNSDBQ_SYSTEM in the FILES section above.

22.8 EXIT STATUS

Success (exit status zero) occurs if a connection could be established to the back end database server, even if no records matched the search criteria. Failure (exit status nonzero) occurs if no connection could be established, perhaps due to a network or service failure, or a configuration error such as specifying the wrong server hostname.

22.8.1 SEE ALSO

- `dnsdbq(1)`, `jq(1)`, `libcurl(3)`

23 DNSDB search glob guide

Globbering is an advanced form of wildcard searches, more powerful than DNSDB's Standard Search left-hand or right-hand wildcards, but not as advanced as Farsight Compatible Regular Expressions (FCRE). They can be simpler to write, especially for API users who are not familiar with regular expressions.

In general, Farsight's glob implementation follows standard [Unix glob\(7\) semantics](#), but not what's sometimes referred to as "extended globbing."

Glob searches are evaluated against the DNS master file form of the hostnames (aka rnames) and rdata values, which by design contains only printable ASCII characters. All non-printable characters, including octets outside the ASCII range, are converted to escape sequences in the form `\DDD` (backslash followed by three decimal digits) per [RFC 1035](#). This is only applicable to RData (RHS) queries.

23.1 Glob Syntax

A glob is a string of printable characters with the following characters given special meaning:

- `*` -- Match any zero or more characters.
- `?` -- Match exactly any one character.
- `[` -- Begin a character class. Any of the contained characters or ranges will match.
- `]` -- End a character class.
- `\` -- Escape the next character (but not within a character class)

Any other characters in globbing pattern get matched exactly as written, except that characters are *not* case sensitive.

23.1.1 Character Class Syntax

A character class is a set of characters enclosed between an opening `[` and a closing `]`. A simple example is `[m-z1-3]` to match characters m through z and 1 to 3.

Within the character class, the following characters are handled specially:

- `!` -- If the first character after the opening `[`, denotes a negated character class, i.e. a class which matches any character not listed in the remainder of the class.
- `]` -- If the first character after the opening `[` or `[!`, encodes a literal `]` as a member of the class. `A]` after the first character after the opening `[` or `[!` ends the character class.
- `-` -- If the first character after the opening `[` or `[!` or the last character before the closing `]`, encodes a literal `-` as a member of the character class.
 - If between two characters A and B, encodes the range of characters between A and B, inclusive, as members of the character class. The character A must occur before B in ASCII encoding.

The sequences `[.` and `[=` are not allowed between the opening `[` or `[!` and the closing `]`, to prevent confusion with unsupported POSIX collation sequences and collation classes.

If the sequence `[:` appears in a character class, it must be the beginning of one of the following *POSIX character classes*:

- `[:a\lum:]` (POSIX character class) -- Alphanumeric characters 0-9, A-Z, and a-z
- `[:a\pha:]` (POSIX character class) -- Alphabetic characters A-Z, a-z

- `[:blank:]` (POSIX character class) -- Blank characters (space and tab)
 - Only printable characters occur in searchable strings and space is the only printable whitespace character, thus use of `[:blank:]` is equivalent to a space character.
 - Tabs in data appear as the escape sequence `\009` and can be matched with the glob pattern `\\009`.
- `[:cntrl:]` (POSIX character class) -- Control characters
 - Only printable characters occur in searchable strings, so `[:cntrl:]` will not match any characters.
 - Control characters in data will appear as escape sequences in the form `\DDD` (backslash followed by three digits). To match one of those, you need to escape the backslash with another backslash. For example, to match the literal string `\004`, use the glob pattern `\\004`.
- `[:digit:]` (POSIX character class) -- Decimal digits 0-9
- `[:graph:]` (POSIX character class) -- Any printable character other than space.
 - Only printable characters occur in searchable strings, thus a character class containing `[:graph:]` is equivalent to `[!]` (negated character class containing only a space).
- `[:lower:]` (POSIX character class) -- Lower case alphabetic characters a-z
 - Hostnames will be folded to lower case, thus use of `[:lower:]` is equivalent to `[:alpha:]`.
- `[:print:]` (POSIX character class) -- Any printable character
 - Only printable characters occur in searchable strings, so `[:print:]` will match any character.
- `[:punct:]` (POSIX character class) -- Punctuation characters (printable characters other than space and `[:alnum:]`)
- `[:space:]` (POSIX character class) -- Any whitespace character
 - The space character is the only printable whitespace character, thus use of `[:space:]` is equivalent to a space character.
- `[:upper:]` (POSIX character class) -- Upper case alphabetic characters A-Z
 - Since all of our data is indexed as lower-case, this is not useful as it is equivalent to `[:lower:]`.
- `[:xdigit:]` (POSIX character class) -- Hexadecimal digits 0-9, a-f, A-F

The above named character classes must appear inside an enclosing `[` and `]`, e.g. `[:digit:][:punct:]` to match a digit or punctuation character. Without the enclosing braces, `[:digit:]` will match the characters `:`, `d`, `i`, `g`, or `t`.

Neither the above character classes nor a character range may begin or end a character range. For example, the character class expressions `[0-[:alpha:]]` and `[a-n-z]` are invalid.

All other characters between the opening `[` or `[!` and the closing `]` are added to the character class, including the backslash `\` character.

There is no way to express a character class containing a single `!` character.

23.1.2 Important notes

- Glob searches are not case sensitive.
- Globbing patterns are “anchored” front and back by default. (This is a major difference from FCRE.)

- All hostnames (rrnames) in the DNS dataset end in a ., which must be accounted for in globs.
 - Therefore, a search for *.com will not match any hostnames. A glob that searches in rrnames **must** end in something that matches a ., so *.com. would match what was intended.
- All well-formed rdata we currently index in the DNS dataset ends in a . or a ", which should be accounted for in globs.
 - Therefore, a glob that searches in rdata *should* end in something that matches a . or a ".
- There must be at least two consecutive non-wildcard characters in the pattern. The implicit front and back anchor counts as a non-wildcard character.

23.2 Examples

- To match hostnames with a label containing the word “smoke”:
 - Glob pattern: *smoke*
 - Search type: rrnames search
 - Example results:
 - * smokeping.pdf.ac.
 - * smoke.tesla.ac.
- To match hostnames with a label containing the word “cider” but not containing “hard”:
 - Glob pattern: *cider*
 - Search type: rrnames search with exclude filter *hard*
 - Example results:
 - * ciderpress.ca.
 - * colombus.citycider2018.eventbrite.ca.
- To match hostnames with a label ending in “www.” and a later label starting with “.com”:
 - Glob pattern: *www.*.com*
 - Search type: rrnames search
 - Example results:
 - * www.example.com.
 - * dev-www.subdomain.example.com.
 - * www.example.com.cdn.net.
 - * stage-www.dev.community.org.
- To match hostnames starting with “www.” and ending in “.com.”:
 - Glob pattern: www.*.com.
 - Search type: rrnames search
 - Example results:
 - * www.example.com.
 - * www.subdomain.example.com.
- To match hostnames starting with “www.” and ending with “.com” with no other dots in between:
 - This *cannot* be done in a general way using globs; use regular expression instead.
- To match hostnames starting with “www” optionally preceded by a “dev-” or “stage-” prefix in a .net or .edu domain:
 - This *cannot* be done in a general way using globs; use regular expression instead.

-
- To match TXT records encoding an SPF policy with a ~all default:
 - Glob pattern: "v=spf1 * ~all"
 - Search type: rdata search
 - Example results:
 - * "v=spf1 a mx ~all"
 - * "v=spf1 a 10.2.0.0/16 ~all"
 - To match single character domain names (which are really two character domain names when you add the implicit trailing '.'):
 - Glob pattern: ?.
 - Search type: rnames or rdata search
 - Example results:
 - * a.
 - * 0.
 - To match "bri" followed by exactly any three characters followed by "morning" followed by anything (or nothing):
 - Glob pattern: bri???morning*
 - Search type: rnames search
 - Note: A question mark matches exactly one character
 - Example results:
 - * brightmorning.com
 - * brightmorningtoday.com
 - To match "ns" followed by any single digit followed by anything (or nothing) and ending in ".net.":
 - Glob pattern: ns[0-9]*.net.
 - Search type: rnames search
 - Example results:
 - * ns0.fsi.net
 - * ns0abc.fsi.net

24 DNSDB Search with Compatible Regular Expressions

DNSDB Farsight Compatible Regular Expressions (FCRE) provides regular expression (regexp) functionality for searching Domain Name System (DNS) hostnames and rdata values in DNSDB. The system evaluates regexp searches against the DNS master file form of the hostnames and rdata values, which by design contains only printable ASCII characters. The system converts all non-printable characters, including octets outside the ASCII range, to escape sequences in the form `\DDD` (backslash followed by three decimal digits) per [RFC 1035](#). This is only applicable to RData (RHS) queries.

For this limited use case, DNSDB FCRE provides a simplified subset of the Portable Operating System Interface (POSIX) Extended Regular Expression syntax, with the most notable restrictions being:

1. Only printable characters are allowed in a regexp.
2. Hexadecimal or octal escape sequences aren't allowed in a regexp.
3. Only special characters may be escaped with `\`. Note that `]` and `}` aren't considered special characters, but `[` and `{` are.
4. POSIX collating elements (e.g., `[=ch=]`, `[.a.]`) in character classes aren't supported. The sequences `[=` and `[.` aren't allowed in character classes.
5. As in POSIX regexps, the character `\` has no special meaning within a character class, so the class `[w]` matches the characters `\` or `w`.
6. Capturing groups and backreferences aren't supported.

Note that restriction (3) means that Perl Compatible Regular Expressions (PCRE) extensions such as `\w` (word characters) and `\d` (digits) aren't allowed in FCRE regexps.

24.1 Regexp syntax

A regular expression is a string of printable characters. The following characters have special meaning:

- `\` -- Escape the next character, which must be a special character. A regexp may not end with an unescaped backslash, or contain an unescaped backslash followed by a character other than another backslash or the special characters listed below, except inside of a character class.
- `^` -- Matches the beginning of the subject string.
- `$` -- Matches the end of the subject string.
- `[` -- Begin a character class
- `.` -- A special character class matching any character.
- `(` -- Begin a sub-pattern. Sub-patterns may occur within other sub-patterns.
- `)` -- End a sub-pattern.
- `|` -- Specify an alternative match. A pattern or subpattern matches if the pattern before or after the `|` matches.
- `*` -- Match the previous character, character class, or subpattern zero or more times.
- `?` -- Match the previous character, character class, or subpattern at most once.
- `+` -- Match the previous character, character class, or subpattern at least once.
- `{` -- If followed by a character other than a decimal digit, is treated as a literal `{` character. Such a `{` may be escaped with a backslash even though it isn't technically a special character in this context.

If followed by a decimal digit, begins a bounded match specification. "{n}" matches exactly n repetitions of the previous character, character class, or subpattern. "{n,m}" with $m \geq n$ matches at least n but at most m repetitions.

24.2 Character class syntax

A character class is a set of characters enclosed between an opening [and a closing]. Within the character class, the following characters have special handling:

- ^ -- If the first character after the opening [, denotes a negated character class, i.e. a class which matches any character not listed in the remainder of the class
-] -- If the first character after the opening [or [^, encodes a literal] as a member of the class. A] after the first character after the opening [or [^ ends the character class.
- - -- If the first character after the opening [or [^ or the last character before the closing], encodes a literal - as a member of the character class. If between two characters A and B, encodes the range of characters between A and B, inclusive, as members of the character class. The character A must occur before B in ASCII encoding.

The sequences [. and [= aren't allowed between the opening [or [^ and the closing], to prevent confusion with unsupported POSIX collation sequences and collation classes.

If the sequence [: appears in a character class, it must be the beginning of one of the following *POSIX character classes*:

- [:alnum:] (POSIX character class) -- Alphanumeric characters 0-9, A-Z, and a-z
- [:alpha:] (POSIX character class) -- Alphabetic characters A-Z, a-z
- [:blank:] (POSIX character class) -- Blank characters (space and tab)
 - Only printable characters occur in searchable strings and space is the only printable whitespace character, thus use of [:blank:] is equivalent to a space character.
 - Tabs in data appear as the escape sequence `\009` and can be matched with the pattern `\009`.
- [:cntrl:] (POSIX character class) -- Control characters
 - Only printable characters occur in searchable strings, so [:cntrl:] won't match any characters.
 - Control characters in data will appear as escape sequences in the form `\DDD` (backslash followed by three digits). To match one of those, you need to escape the backslash with another backslash. Use the pattern `\[:digit:]{3}` in a regular expression.
- [:digit:] (POSIX character class) -- Decimal digits 0-9
- [:graph:] (POSIX character class) -- Any printable character other than space.
 - Only printable characters occur in searchable strings, thus a character class containing [:graph:] is equivalent to [^] (negated character class containing only a space).
- [:lower:] (POSIX character class) -- Lower case alphabetic characters a-z
 - Hostnames will be folded to lower case, thus use of [:lower:] is equivalent to [:alpha:].
- [:print:] (POSIX character class) -- Any printable character
 - Only printable characters occur in searchable strings, so [:print:] matches any character.
- [:punct:] (POSIX character class) -- Punctuation characters (printable characters other than space and [:alnum:])

- `[:space:]` (POSIX character class) -- Any whitespace character (tab, newline, vertical tab, form feed, carriage return, and space)
 - The space character is the only printable whitespace character, thus use of `[:space:]` is equivalent to a space character.
 - Tabs in data appear as the escape sequence `\009` and can be matched with the pattern `\009`. The other characters can also be matched by searching for their decimal equivalent.
- `[:upper:]` (POSIX character class) -- Upper case alphabetic characters A-Z
 - Since all of our data is indexed as lower-case, this isn't useful as it is equivalent to `[:lower:]`.
- `[:xdigit:]` (POSIX character class) -- Hexadecimal digits 0-9, a-f, A-F

The above named character classes must appear inside an enclosing `[and]`, e.g. `[:digit][:punct:]` to match a digit or punctuation character. Without the enclosing braces, `[:digit:]` will match the characters `:`, `d`, `i`, `g`, or `t`.

Neither the above character classes nor a character range may begin or end a character range. For example, the character class expressions `[0-[:alpha:]]` and `[a-n-z]` are invalid.

All other characters between the opening `[` or `[^` and the closing `]` are added to the character class, including the backslash `\` character.

There is no way to express a character class containing a single `^` character: an escaped `\^` should be used instead of a character class.

24.3 Important notes

- Regular expression searches aren't case sensitive.
- Regular expression patterns are *not* "anchored" front and back by default. (This is a major difference from glob searches.)
- To exactly match a literal `.` (such as between labels in a DNS name), you need to escape it with a backslash. Example pattern: `google\.com`. This isn't necessary if the `.` is inside a character class, for example `foo[.__]bar`. If you don't escape the `.`, the pattern `google.com` will match `'googlexcom'`, `'google_com'`, etc.
- All rnames (i.e. hostnames) in the DNS dataset end in a `.`, which must be accounted for in regular expressions.
- All well-formed rdata we currently index in the DNS dataset ends in a `.` or a `"`, which should be accounted for in regular expressions.
- Wildcard operators (`.`, `*`, `+`, `?`, and character classes like `[a-z]`) must have at least two non-wildcard characters immediately before or after them. Anchors (`^` and `$`) count as non-wildcard characters for this purpose.

24.3.1 Valid wildcard patterns

- Pattern: `^vpr-[a-z]+\.\.*$` - **Valid:** The `$` anchor provides a second non-wildcard character after `.*`
- Pattern: `^vpr-[a-z]+\.\.{2,}` - **Valid:** Explicit 2+ character match after the dot
- Pattern: `example\.com.*` - **Valid:** "com" provides 3 characters before the wildcard
- Pattern: `^www\.\.*\.\.com\.$` - **Valid:** Anchors and literal characters satisfy the requirement

24.3.2 Invalid wildcard patterns

- Pattern: `^vpr-[a-z]+\.\.*` - **Invalid:** Only one character (the escaped dot) before the final `.*` wildcard

- Pattern: `x.*` - **Invalid:** Only one character before the wildcard
- Pattern: `.*y` - **Invalid:** Only one character after the wildcard
- Pattern: `.*example` - **Invalid:** Only one character (the start anchor is implicit but not present)

24.4 Examples

The following examples show regular expression patterns and some of their matching values:

- Pattern: `www\.*\.com`
 - Matches: Hostnames with a label ending in “www.” and a later label starting with “.com”
 - Example results:
 - * `www.example.com.`
 - * `dev-www.subdomain.example.com.`
 - * `www.example.com.cdn.net.`
 - * `stage-www.dev.community.org.`
- Pattern: `^www\.*\.com`
 - Matches: Hostnames starting with “www.” and ending in “.com”
 - Example results: No results
 - * Note: Hostnames in the DNS dataset contain a trailing “.”, which must be accounted for in regexps. This pattern is missing the trailing dot.
- Pattern: `^www\.*\.com\.$`
 - Matches: Hostnames starting with “www.” and ending in “.com.”
 - Example results:
 - * `www.example.com.`
 - * `www.subdomain.example.com.`
- Pattern: `^www\.[^.]+\.com\.$`
 - Matches: Hostnames starting with “www.” and ending with “.com” with no other dots in between
 - Example results:
 - * `www.example.com.`
 - * `www.other-domain.com.`
- Pattern: `^((dev|stage)-)?www\.[^.]+\.(net|edu)\.$`
 - Matches: Hostnames starting with “www” optionally preceded by a “dev-” or “stage-” prefix in a .net or .edu domain
 - Example results:
 - * `www.college.edu`
 - * `dev-www.isp.net`
- Pattern: `^"v=spf1 .* ~all"$`
 - Matches: TXT records encoding an SPF policy with a ~all default
 - Example results:
 - * `"v=spf1 a mx ~all"`
 - * `"v=spf1" " a " "10.2.0.0/16" " ~all"`
- Pattern: `(^|[-_])star([-_]?z[-_]`

-
- Matches: Hostnames that start with “star”, or have “star” as a label or otherwise separate from other letters/digits, followed by an optional dash or underscore, then a z, then a period, dash or underscore
 - Use case: Looking for a visibly embedded trademark
 - Example results:
 - * star-z.at.
 - * edge-star-z-mini-shv-02-mia3.goldmansachs.de.
 - * starz.webex.com.
 - * shooting-starz.tv.

25 DNSDB GitHub repositories

Farsight Security maintains a collection of open-source tools and libraries for working with DNSDB and passive DNS data. These resources are designed for developers building applications that integrate with or process DNSDB data.

25.1 Core Libraries

- **nmsg** - Network message encapsulation library (C)
- **mtbl** - Immutable sorted string table library (C)
- **go-nmsg** - Golang implementation of the nmsg network message encapsulation library
- **golang-framestream** - Frame Streams implementation in Go

25.2 Python Extensions

- **pydnstable** - Python extension module for libdnstable
- **pywdns** - Python extension module for libwdns
- **pymtbl** - Python extension module for libmtbl
- **pynmsg** - Python extension module for nmsg

25.3 Utilities

- **dnstable-convert** - Conversion utility for passive DNS data
- **nmsg-relay** - Utility for uploading NMSG data using the sielink protocol
- **sielink** - Protocol library for uploading data to the Farsight Security Information Exchange
- **dnstap-sensor** - Dnstap-based SIE DNS sensor
- **go-config** - Useful golang types for JSON and YAML config files

25.4 All Repositories

For the complete list of repositories and additional tools, visit:

<https://github.com/orgs/farsightsec/repositories>

25.5 Licensing

Most repositories are licensed under the Apache License 2.0 or Mozilla Public License 2.0. Check individual repository licenses for specific terms.