

DomainTools Iris Investigate API

Generated on April 27, 2026

Contents

1	Sample Response	5
2	Iris API Authentication	6
2.1	Overview	6
2.2	Protecting API Credentials	6
2.3	Open Key Authentication	6
2.4	API Key (Header) Authentication	6
2.4.1	Required API Key Authentication Parameters	6
2.4.2	Example Request with API Key Authentication	7
2.5	HMAC Signed Authentication	7
2.5.1	HMAC-Signed Authentication URI/Parameters	7
2.5.2	Creating a HMAC-signed authentication request	7
2.5.3	Security Note	8
3	Iris Rate Limits	9
3.1	Check your rate limits	9
3.2	Free test queries	9
3.2.1	Endpoints with free test queries	9
3.3	Iris Investigate rate limits and quota consumption	9
3.3.1	Duplicate query policies	9
3.3.2	Activities that consume quota	10
3.4	Iris Enrich rate limits	10
3.5	Iris Detect rate limits	11
3.5.1	Domain endpoints rate limits	11
3.5.2	Development and testing	11
3.5.3	Other endpoints	11
4	Iris API Error Codes	12
4.1	Status Codes	12
4.2	Error Response Format	12
4.3	Common Error Scenarios	13
4.3.1	Authentication Errors (401)	13
4.3.2	Authorization Errors (403)	13

4.3.3	Not Found Errors (404)	13
4.3.4	Rate Limiting (503)	13
4.4	Best Practices	13
4.5	Iris-Specific Notes	14
4.5.1	Partial Content (206)	14
4.5.2	Detect Endpoints	14
4.5.3	Validation Errors (422)	14
5	Iris Investigate API	16
5.1	Overview	16
5.2	Documentation Structure	16
5.3	API Reference	16
5.4	Next Steps	16
6	Iris Investigate API quick start	17
6.1	API Endpoint	17
6.2	API Authentication	17
6.3	Query Limits and Testing	17
6.4	Retrieve Domain Records with a Test Query	17
6.5	Search for a Set of Domain Names	18
6.6	Next Steps	18
7	Iris Investigate API concepts	20
7.1	Key Concepts	20
7.1.1	Guided Pivots	20
7.1.2	Pagination	20
7.1.3	Quota Consumption	20
7.2	Related Topics	20
8	Iris Investigate API guided pivots	21
8.1	How Guided Pivots Work	21
8.2	Example Response	21
8.3	Using Pivot Counts	21
8.4	Pivot Strategies	21
8.5	See Also	22
9	Iris Investigate API pagination	23
9.1	Query Limits	23
9.1.1	Too Many Results	23
9.1.2	Paginated Results	23
9.2	Using the <code>positionParameter</code>	24
9.3	Using the <code>nextParameter</code>	24
9.4	Using <code>page_size</code> to Limit Page Size	24
9.5	Sorting Results	24
9.5.1	Using <code>sort_by</code>	25
9.5.2	Using <code>sort_direction</code>	25
9.6	See Also	25
10	Iris Investigate API quota consumption	26
10.1	Duplicate Query Policy for API	26
10.2	Activities That Consume Quota	26
10.3	Activities That Do Not Consume Quota	26
10.4	Additional Information	26
10.5	See Also	26
11	Iris Investigate API search	27

11.1 Search Overview	27
11.1.1 Example: Search by IP Address	27
11.1.2 Combining Search Parameters	27
11.2 Search Types	27
11.2.1 Base Search Parameters	27
11.2.2 Filter Parameters	27
11.2.3 RDAP and WHOIS Search	28
11.3 Search Examples	28
11.3.1 Example: Filter by TLD	28
11.3.2 Example: Search by Email Domain	28
11.4 See Also	28
12 Iris Investigate API base search parameters	29
12.1 Parameter Reference	29
12.2 Usage Examples	32
12.2.1 Search by IP Address	32
12.2.2 Search by Email Domain	32
12.2.3 Search by SSL Hash	32
12.2.4 Combine Multiple Parameters	33
12.3 See Also	33
13 Iris Investigate API filter parameters	34
13.1 Parameter Reference	34
13.2 Usage Examples	35
13.2.1 Filter by TLD	35
13.2.2 Filter by Create Date	36
13.2.3 Filter by Risk Score	36
13.2.4 Filter by Active Status	36
13.2.5 Filter by First Seen	36
13.3 See Also	36
14 Iris Investigate API RDAP and WHOIS search	37
14.1 Search Within Parsed Fields	37
14.2 Compare WHOIS and Domain RDAP Records	37
14.2.1 Example: Request Parsed Domain RDAP	37
14.2.2 Response Structure	37
14.3 Use Cases	38
14.3.1 Compare Registration Data Sources	38
14.3.2 Search Historical WHOIS	38
14.4 See Also	39
15 Work with domain tags in the Iris Investigate API	40
15.1 Use Domain Tags as a Search Parameter	40
15.1.1 Examples	40
15.2 Use Domain Tags as a Search Filter	40
15.2.1 Examples	41
15.3 Tag Parameters Reference	41
15.4 See Also	41
16 Retrieve screenshots with the Iris Investigate API	42
16.1 Step 1: Retrieve the Screenshot URL	42
16.1.1 Example Queries	42
16.1.2 Query Parameters	42
16.1.3 Response Fields	42
16.1.4 Response Example	43

16.2 Step 2: Retrieve and Optionally Resize the Screenshot Image	43
16.2.1 Query Parameters	43
16.2.2 Example Query	44
16.3 Interpreting Screenshot Metadata	44
16.3.1 Initial Capture	44
16.3.2 Gather Attempt Was Duplicate of Initial Screenshot	44
16.3.3 Recapture Failed After Previous Successful Capture	45
16.3.4 Duplicate Capture Followed by Failed Attempt	45
16.4 See Also	46
17 Monitor newly active domains with the Iris Investigate API	47
17.1 How It Works	47
17.2 Implementation Considerations	47
17.2.1 Basic Monitoring Pattern	47
17.2.2 Choosing Time Parameters	47
17.2.3 Handling Large Result Sets	47
17.2.4 Quota Optimization	48
17.3 Use Cases	48
17.4 Examples	48
17.4.1 Monitor Domains on a Specific IP	48
17.4.2 Monitor Domains with Specific SSL Certificate	48
17.4.3 Monitor Domains on Nameserver with TLD Filter	48
17.5 See Also	49
18 Integrate with Iris Investigate UI	50
18.1 When to Use UI Integration	50
18.2 How It Works	50
18.3 Using the search_hash Parameter	50
18.3.1 Example Query	50
18.3.2 Example with cURL	50
18.4 Workflow	51
18.5 Benefits	51
18.6 See Also	51
19 IrisQL: Iris Query Language for API	52
19.1 Using IrisQL	52
19.2 Getting started	52
19.2.1 Endpoint	52
19.2.2 Authentication	52
19.3 Request format	52
19.3.1 Basic structure	52
19.3.2 Version comment	52
19.3.3 Query parameters	53
19.4 Response format	53
19.4.1 Response structure	53
19.5 Examples	54
19.5.1 Basic query	54
19.5.2 Complex query with multiple conditions	54
19.5.3 Query with sorting and pagination	54
19.5.4 Query with XML response format	54
19.6 IrisQL syntax reference	55
19.6.1 Query structure	55
19.6.2 Field names and API parameters	55
19.6.3 Supported fields	55
19.6.4 Operators	59

19.6.5 Searching by field presence vs. specific values	60
19.6.6 Logical operators	61
19.6.7 Registration data sources	62
19.6.8 Historical search syntax	62
19.6.9 Data types	63
19.7 Rate limits	64
19.8 See also	65

The Account Information API provides a quick and easy way to get a snapshot of API product usage for an account. Usage is broken down by day and by month.

```
https://api.domaintools.com/v1/account/
```

1 Sample Response

```
{
  "response": {
    "account": {
      "api_username": "domaintools-api-account",
      "active": true
    },
    "products": [
      {
        "id": "domain-profile",
        "per_month_limit": "100000",
        "per_minute_limit": "120",
        "absolute_limit": "1000",
        "usage": {
          "today": 5,
          "month": 152
        },
        "expiration_date": "2020-01-01"
      }
    ]
  }
}
```

2 Iris API Authentication

2.1 Overview

Most requests sent to the DomainTools API require authentication. We support two authentication schemes with different levels of security.

We recommend the following practices for authentication:

- Authentication with HTTPS (HTTP over SSL) instead of HTTP.
- Using HMAC signed queries or Header Authentication instead of Open Key Authentication.
- Using SHA-256 HMAC for the signed queries instead of MD5 or SHA-1.

2.2 Protecting API Credentials

Your account will be charged for all queries authenticated with your username and key, even if you later determine the requests were fraudulent or its use unauthorized. We recommend the following steps to protect your API credentials:

- For support requests, do not send the full API key to DomainTools support.
- Use a HMAC-signed approach to requests. This ensures that the user and token are not sent as part of the URL.
- When required, reset the API key.
- Do not place them into a public repository, such as a git repository.

2.3 Open Key Authentication

This is the easiest authentication scheme to implement, but also the most insecure. You should take precautions in the design of your application to ensure your key is not compromised.

```
https://api.domaintools.com/v1/domaintools.com?api_username=example&api_key=xxxxxx
```

Required Parameters	Value
api_username	Your API username
api_key	Your API key

2.4 API Key (Header) Authentication

Authenticate your requests by including the API key in the header of each HTTP request. The API key serves as a unique identifier and is used to authenticate your requests.

2.4.1 Required API Key Authentication Parameters

Required Parameters	Value
X-Api-Key	MY_API_KEY

2.4.2 Example Request with API Key Authentication

```
curl -H 'X-API-Key: MY_API_KEY'
'https://api.domaintools.com/v1/feed/nod/?after=-60'
```

2.5 HMAC Signed Authentication

HMAC, or hashed message authentication code, is our preferred authentication scheme. It follows the principles outlined in RFC2104 and provides a straightforward but secure method of protecting your API key.

It involves passing a hash composed of your api username, the current date and time, and the request URI. This hash is then signed with your authentication key. The result is a request that expires after a brief period of time and, most importantly, does not contain your authentication key.

MD5, SHA-1, and SHA-256 are supported HMAC hashing algorithms. We recommend SHA256 HMAC, as a successor of SHA1. Although NIST policy on hash functions still allows HMAC SHA-1, see [Hash Functions in the NIST CSRC](#) page. Overall, SHA-256 HMAC is also a more secure (ex, against brute force attacks) alternative to MD5 HMAC.

Please check your own compliance obligations, or guidance, as to what cryptographic functions should be used. For example, see the [NIST FIPS 140-2 Guidance](#) for more information for organizations that must use FIPS 140-2 compliant algorithms.

2.5.1 HMAC-Signed Authentication URI/Parameters

Required Parameters	Value
api_username	Your API username
timestamp	Current timestamp, in ISO 8601 format. Timestamps should contain the timezone offset, like below: 2020-02-01T14:37:59-0800 2020-02-01T22:37:59Z 2020-02-02T10:37:59+1200
signature	HMAC signature of your request. SHA-256 HMAC is recommended. MD5 HMAC and SHA-1 HMAC are supported. Only sign the URL path.

2.5.2 Creating a HMAC-signed authentication request

To create the signature, build a string with your API username, the timestamp in ISO 8601 format, and the URI you are requesting. That string is then signed with your API key using an HMAC function to generate the signature. The steps are below.

1. Add your api_username.
The API username is the account that has been provisioned access to the relevant DomainTools API product (aka API endpoint). If this username does not have access rights, an error message will be returned.
2. Associate a timestamp.
The timestamp value for the hash and timestamp parameter need to be identical. Make sure your server time is accurate and always use a fresh timestamp.

3. Add the request URI.
Please see the individual API product User Guides for the endpoint's URI format.
4. Add the host.
The host is the API endpoint server. In this case, it is api.domaintools.com.
5. Use the appropriate HMAC algorithm.
MD5, SHA-1, or SHA-256 for the hashing algorithm are supported. See the HMAC Algorithms section for more information.

The result is a request that expires after a brief period of time and, most importantly, does not contain your authentication key in the clear.

The resulting URLs will generally be similar, regardless of whichever programming language is used.

2.5.3 Security Note

Ensure that your current language version supports the required HMAC (and other relevant) extensions and libraries. For example, a failure in HMAC-signed URL will result in a URL with no signature value, such as below:

```
http://api.domaintools.com/v1/yourdomain.com/whois
```

3 Iris Rate Limits

This document explains rate limits and quota consumption for Iris Investigate, Iris Enrich, and Iris Detect in both the API and UI.

3.1 Check your rate limits

Use the following methods to determine the current rate limit associated with Iris Investigate, Iris Enrich, and Iris Detect API endpoints:

1. /account Endpoint

- Querying the Account Information endpoint provides details on query limits, usage, and expiration dates for all licensed Iris endpoints.
- For more information, refer to the [Account Information endpoint documentation](#).

2. API Admin Dashboard

- The API Admin can access their API dashboard to view query limits, usage, and expiration dates for all licensed endpoints.
- To access the API dashboard, visit the [DomainTools research page](#), select the account dropdown menu, and click on API Admin.

3.2 Free test queries

Multiple endpoints, including Iris Detect and Investigate, offer free test queries that don't count against your quota. Use these domains for testing and development:

- [domaintools.com](#)
- [dnsdb.info](#)
- [example.com](#)

3.2.1 Endpoints with free test queries

The following endpoints support free test queries with the domains listed above:

[Domain Profile](#), [Domain Reputation](#), [Hosting History](#), [Iris Enrich and Investigate](#), [Reverse IP](#), [WHOIS history](#), [WHOIS Lookups](#)

Using these test domains allows you to verify API integration and test functionality without consuming your quota.

3.3 Iris Investigate rate limits and quota consumption

The system measures quotas at the group level and resets them each month.

3.3.1 Duplicate query policies

Iris Investigate treats identical queries differently depending on whether you use the UI or API:

3.3.1.1 User Interface

Identical queries within 30 days don't count against your quota. This includes:

- All searches (omnisearch and advanced search)
- Search hash queries
- Search hash reloading
- Search node revisits

A query is considered identical if it has the same filters, sorting, and pagination parameters.

3.3.1.2 API

Regular queries: Identical queries within 1 hour don't count against quota. A query is considered identical if it has the same filters, sorting, and pagination parameters as a previous query made within the last hour.

Search hash queries: Identical search hash queries within 10 minutes don't count against quota. A search hash query uses the `search_hash` parameter to retrieve previously saved search results.

3.3.2 Activities that consume quota

3.3.2.1 Pivot Engine Queries in the Iris Investigate UI

The following activities consume your quota:

- Executing an omnisearch (from landing or search pages) that returns results
- Executing an advanced search that returns results
- Sending a result to the Pivot Engine (including `narrow`, `expand`, `new`, and `exclude` functions)
- Revisiting a search node more than 30 days since it was created
- Loading new pages in the Pivot Engine
- Sorting Pivot Engine results

For information on duplicate queries that don't consume quota, see [Duplicate query policies](#).

3.3.2.2 Passive DNS (pDNS) Queries in the Iris Investigate UI

The following activities consume your quota:

- Executing a search query that returns results (from either the search field or popovers throughout the UI)
- Executing/triggering the load more (infinite scroll) function in search results
- Revisiting a search node more than 30 days since it was created

For information on duplicate queries that don't consume quota, see [Duplicate query policies](#).

3.3.2.3 Queries in the Iris Investigate API

The following activities consume your quota:

- Executing a query
- Loading additional pages of results

For information on duplicate queries that don't consume quota, see [Duplicate query policies](#).

3.4 Iris Enrich rate limits

Iris Enrich uses an independent service level with its own rate limits, separate from Iris Investigate. This independence allows Iris Enrich to be used at much greater scale and throughput than Iris Investigate.

Key differences:

- **Independent quota:** Enrich queries don't count toward your Iris Investigate quota
- **Separate service level:** Rate limits are defined independently for Enrich
- **Higher throughput:** Optimized for high-volume domain enrichment use cases
- **No pivot parameters:** Unlike Investigate, Enrich is optimized for straightforward domain enrichment rather than pivoting

Your Enrich service level, query caps, and rate limits are configured independently on your enterprise account. For complete details on using Iris Enrich, consult the [Iris Enrich API documentation](#).

3.5 Iris Detect rate limits

Iris Detect has specific endpoint rate limits designed for monitoring and triage workflows.

3.5.1 Domain endpoints rate limits

The domains endpoints (/domains/new and /domains/watched) have a rate limit of **1 query per hour** for each endpoint.

Pagination: Requests using the offset parameter to retrieve additional pages of results aren't restricted by the hourly limit. If your hourly query returns more domains than the response limit, you can retrieve the complete result set using pagination without waiting.

3.5.2 Development and testing

During integration and testing, use the preview parameter to bypass rate limits:

- Including preview=1 in requests limits responses to 2 domains
- Allows up to 30 requests per minute
- Enables rapid iteration during development

3.5.3 Other endpoints

Monitor management endpoints and domain triage actions (watchlist, ignore, escalate) have standard API rate limits. For complete details on Iris Detect API endpoints and usage, consult the [Iris Detect API documentation](#).

4 Iris API Error Codes

The Iris API uses standard HTTP status codes to indicate the success or failure of requests. Understanding these codes helps you handle errors gracefully and troubleshoot issues effectively.

4.1 Status Codes

Status Code	Description
200	OK - The request was successful.
206	Partial Content - The request was partially successful. Some data is unavailable but the request returned what was available. This typically occurs when backend services are temporarily unavailable or when data for some requested items cannot be retrieved.
400	Bad Request - The request is malformed or contains invalid parameters. Check the error message for details about what needs to be corrected.
401	Unauthorized - Authentication credentials are missing, invalid, or expired. Verify your API key and authentication method.
403	Forbidden - The authenticated account does not have permission to access this resource or endpoint. This may indicate insufficient subscription level or access rights.
404	Not Found - The requested resource does not exist. This may occur when querying for a domain that has never been registered or for data that is not available in our systems.
500	Internal Server Error - An unexpected error occurred on the server. If this persists, contact DomainTools support at enterprisesupport@domaintools.com .
503	Service Unavailable - The service is temporarily unavailable, typically due to maintenance or high load. Implement exponential backoff and retry the request.

4.2 Error Response Format

Error responses follow a consistent JSON structure:

```
{
  "error": {
    "code": 400,
    "message": "Invalid domain name format"
  }
}
```

```
}  
}
```

4.3 Common Error Scenarios

4.3.1 Authentication Errors (401)

Cause: Missing or invalid API credentials.

Solutions:

- Verify your API key is correct
- Check that your authentication method (header, HMAC, or basic auth) is properly configured
- Ensure your API key hasn't expired

4.3.2 Authorization Errors (403)

Cause: Insufficient permissions or subscription level.

Solutions:

- Verify your account has access to the requested endpoint
- Check your subscription includes the API product you're trying to use
- Contact your account manager if you need additional access

4.3.3 Not Found Errors (404)

Cause: Requested resource doesn't exist.

Common scenarios:

- Querying a domain that has never been registered
- Requesting historical data that predates our records
- Using an invalid endpoint path

4.3.4 Rate Limiting (503)

Cause: Too many requests in a short time period.

Solutions:

- Implement exponential backoff retry logic
- Check your rate limits using the Account Information endpoint
- Distribute requests over time rather than in bursts
- Consider upgrading your subscription for higher rate limits

4.4 Best Practices

1. **Always check status codes:** Don't assume success - verify the response status code
2. **Parse error messages:** Error messages contain specific details about what went wrong
3. **Implement retry logic:** Use exponential backoff for 500 and 503 errors
4. **Log errors:** Keep detailed logs of errors for troubleshooting
5. **Monitor rate limits:** Track your API usage to avoid hitting limits

4.5 Iris-Specific Notes

4.5.1 Partial Content (206)

The Iris Enrich and Investigate endpoints may return a 206 status code when:

- Some backend data sources are temporarily unavailable
- Data for certain requested domains cannot be retrieved
- The response contains partial results rather than complete data

When you receive a 206 response, the returned data is still valid and usable - it simply indicates that some information is missing.

4.5.2 Detect Endpoints

Iris Detect endpoints (watchlists and monitors) use a subset of the standard error codes:

- 200 (OK)
- 400 (Bad Request)
- 401 (Unauthorized)
- 403 (Forbidden)
- 404 (Not Found)
- 422 (Unprocessable Entity) — write endpoints only

These endpoints do not return 500 or 503 errors in normal operation.

4.5.3 Validation Errors (422)

Iris Detect write endpoints (POST, PUT, and PATCH) return a 422 Unprocessable Entity status code when the request body is syntactically valid JSON but contains values that fail server-side validation. Unlike the standard error format, the 422 response includes per-field validation messages:

```
{
  "error": {
    "code": 422,
    "summary": "The given data was invalid.",
    "messages": {
      "term": ["The term field is required."],
      "watchlist_domain_ids": ["The watchlist domain ids field is required."]
    }
  },
  "resources": {
    "support": "https://docs.domaintools.com"
  }
}
```

Common validation triggers include:

- Missing required fields (for example, term when creating a monitor)
- Values that don't meet minimum length requirements (for example, term must be at least 3 characters)
- Invalid enum values (for example, an unrecognized state value)

This applies to the following endpoints:

Endpoint	Method
/v1/iris-detect/domains/	PATCH
/v1/iris-detect/escalations/	POST
/v1/iris-detect/monitors/	POST
/v1/iris-detect/monitors/	PUT

Iris Enrich and Iris Investigate endpoints do not return 422. These endpoints return 400 for invalid parameters.

5 Iris Investigate API

5.1 Overview

The Iris Investigate API enables deep domain analysis and infrastructure mapping at human scale. It provides comprehensive domain profiles with dozens of attributes, pivot searches across multiple data points, and guided pivot counts to identify meaningful connections.

For a complete overview of Iris Investigate capabilities and use cases, see the [Iris Overview](#).

Consult the [OpenAPI reference](#) for the Iris API suite for additional detail.

Key characteristics of the Iris Investigate API include:

- Dozens of domain name attributes on every result, including:
 - Domain risk scores from proximity and threat profile algorithms
 - RDAP, WHOIS, IP, active DNS, website and SSL data
 - Counts of connected domains on most attributes
- Unified API endpoint for both domain profiles and pivot searches
 - Support for batch processing and pagination
 - Search by identity, IP, name server, mail server, SSL certificate, and more
- Offered with interactive Iris web app

5.2 Documentation Structure

This documentation is organized into the following sections:

- [Quick Start](#): Get started with authentication and your first API calls
- [Concepts](#): Understand core concepts like guided pivots, pagination, and quota consumption
- [Search](#): Learn about search parameters and filtering capabilities
- [How-To Guides](#): Step-by-step guides for common tasks

5.3 API Reference

For complete API specifications and interactive documentation:

- [OpenAPI Specification](#) - Complete YAML specification for all Iris APIs
- [SwaggerHub Interactive Docs](#) - Test endpoints and explore the API interactively
- [Authentication](#) - Authentication methods and setup
- [Rate Limits](#) - Quota policies and limits
- [Error Codes](#) - Complete error code reference

5.4 Next Steps

- New to the API? Start with the [Quick Start guide](#)
- Need to understand how pivots work? See [Guided Pivots](#)
- Looking for specific parameters? Check the [Search Parameters](#) documentation

6 Iris Investigate API quick start

This guide will help you make your first API calls to the Iris Investigate API.

6.1 API Endpoint

```
https://api.domaintools.com/v1/iris-investigate/
```

6.2 API Authentication

The Iris APIs share the same [authentication mechanisms](#) as DomainTools Feeds.

Link your credentials to your organization's DomainTools enterprise account, which must be authorized for Iris Investigate.

Use Header Authentication by passing your API key in the X-Api-Key header field (example below).

6.3 Query Limits and Testing

!!!note Test and configure with no-charge test queries.

Use no-charge test queries for configuration without consuming your quota:

- ?domain=domaintools.com
- ?domain=dnsdb.info
- ?domain=example.com
- ?ip=199.30.228.112

For complete details on free test queries and rate limits, see [Iris Rate Limits](#).

6.4 Retrieve Domain Records with a Test Query

Submit a HTTP GET request:

Query URL

```
https://api.domaintools.com/v1/iris-investigate/?domain=domaintools.com
```

cURL Example

```
curl -X GET \  
  'https://api.domaintools.com/v1/iris-investigate/?domain=domaintools.com' \  
  -H 'X-Api-Key: YOUR_API_KEY'
```

The API returns default records for domaintools.com as objects in the results array.

```
{
  "response": {
    "limit_exceeded": false,
    "has_more_results": false,
    "message": "Enjoy your data.",
    "results_count": 1,
    "total_count": 1,
    "results": [
      {
        "domain": "domaintools.com",
        "whois_url": "https://whois.domaintools.com/domaintools.com",
        "adsense": {
          "value": "",
          "count": 0
        },
        ...
      }
    ]
  }
}
```

6.5 Search for a Set of Domain Names

Search for a single domain name:

```
https://api.domaintools.com/v1/iris-investigate/?domain=domaintools.com
```

Pass a comma-separated list of up to 100 domain names in a HTTP GET request:

Query URL

```
https://api.domaintools.com/v1/iris-
↳ investigate/?domain=domaintools.com,domaintools.net,example.com
```

cURL Example

```
curl -X GET 'https://api.domaintools.com/v1/iris-
↳ investigate/?domain=domaintools.com,domaintools.net,example.com' \
-H 'X-Api-Key: YOUR_API_KEY'
```

Consider using a POST request for multiple domains:

```
curl -X POST 'https://api.domaintools.com/v1/iris-investigate/' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'X-Api-Key: YOUR_API_KEY' \
-d 'domain=domaintools.com,domaintools.net,example.com'
```

6.6 Next Steps

- Learn about [search parameters](#) to find domains by IP, email, SSL hash, and more

- Understand [pagination](#) for handling large result sets
- Explore [guided pivots](#) to identify meaningful connections

7 Iris Investigate API concepts

Understanding these core concepts will help you use the Iris Investigate API effectively.

7.1 Key Concepts

7.1.1 Guided Pivots

Learn how guided pivot counts help identify attributes shared with a small number of other domains, making it easier to find related infrastructure.

7.1.2 Pagination

Understand how to handle large result sets using pagination, including the `position`, `next`, `page_size`, and sorting parameters.

7.1.3 Quota Consumption

Learn what counts against your API quota and how the duplicate query policy works to optimize your usage.

7.2 Related Topics

- [Search Parameters](#) - Learn about available search and filter parameters
- [OpenAPI Specification](#) - Complete API specification and interactive documentation

8 Iris Investigate API guided pivots

Guided pivots help identify attributes that are shared with a relatively small number of other domain names. The smaller the count, the more likely the domains are to be related.

8.1 How Guided Pivots Work

The Iris Investigate API delivers these counts for nearly every attribute in a domain response, on every domain record, even when processing a batch of domain names.

These counts are included in the API response as a property of the attribute, adjacent to the attribute value. This also explains why the value of a field is one level deeper than you may expect.

8.2 Example Response

```
ip:
  [
    {
      address:
        { value: "199.30.228.112", count: 3 },
      asn:
        [
          { value: 17318, count: 101 }
        ],
      country_code: { value: "us", count: 239988363 },
      isp: { value: "Domaintools LLC", count: 108 }
    }
  ]
```

In this example, we identify the IP address 199.30.228.112, ASN 17318 and ISP Domaintools LLC as potential pivot points, or at the very least, as meaningful analytics to help profile the domain name. For example, IP addresses with very few other domains pointed to them often represent dedicated hosting controlled by the same entity.

8.3 Using Pivot Counts

In the Iris Investigate UI, we use a default threshold of 500 connections to decide which attributes to draw the user's attention to. Consider starting with a similar threshold for your integration, but provide the user the option to choose a different threshold to match their use case.

Note that empty data elements will have a count of 0.

8.4 Pivot Strategies

When analyzing pivot counts:

1. **Low counts (< 100):** Strong indicators of related infrastructure
2. **Medium counts (100-500):** Potentially meaningful connections worth investigating
3. **High counts (> 500):** Common attributes that may not indicate direct relationships

8.5 See Also

- [Search Parameters](#) - Use pivot insights to craft targeted searches
- [Monitor Domains](#) - Monitor newly active domains with specific attributes

9 Iris Investigate API pagination

Learn how to handle large result sets using pagination parameters.

9.1 Query Limits

9.1.1 Too Many Results

The API returns domains only if there are fewer than 10,000 matching domains. [Use filter parameters](#) to narrow your results.

Searches with >10,000 results return a HTTP 200 response and code: 413 in the response body. For example, this search for a common nameserver_domain:

```
https://api.domaintools.com/v1/iris-investigate/?nameserver_domain=markmonitor.zone
```

Returns:

```
{
  "response": {
    "error": {
      "code": 413,
      "message": "More than 10000 matched - you may need to refine your query."
    },
    "limit_exceeded": true,
    "has_more_results": true,
    "message": "Maximum 10000 returned - you may need to refine your query.",
    "missing_domains": []
  }
}
```

9.1.2 Paginated Results

Queries with more than 500 and fewer than 10,000 results return HTTP 200 with the partial results, with a default response of 500. For example:

```
https://api.domaintools.com/v1/iris-investigate/?redirect_domain=domaintools.com
```

This response snippet shows how it includes both "limit_exceeded": false and "has_more_results": true, which means that the request is within your response limit, and more results are available. It also returns the position field to use for pagination.

```
{
  "response": {
    "limit_exceeded": false,
    "has_more_results": true,
    "message": "There is more data for you to enjoy.",
    "results_count": 500,
    "total_count": 1381,
    "position": "f825bb76cd46471482e016f944b1131d",
    "results": [
      {
```

```
"domain": "vilamoda.com",  
...
```

9.2 Using the position Parameter

Note that `position` is both the field name, and the query parameter name. Using the `position` **string** from the response object above, we page to the next 500 results:

```
https://api.domaintools.com/v1/iris-investigate/?redirect_  
↳ domain=domaintools.com&position=f825bb76cd46471482e016f944b1131d
```

The final page of results is indicated by `has_more_results` flipping to `false` and a `results_count`: `<500`.

9.3 Using the next Parameter

Include `next=true` for `>500` results (with the default 500 response page size) and the response will include a `next` field. The `next` field appears below `position` and contains the URL of the API call for the next page of results (including your query parameters).

A request for `https://api.domaintools.com/v1/iris-investigate/?redirect_domain=domaintools.com&next=true` returns a response with the `next` field:

```
{  
  "response": {  
    "limit_exceeded": false,  
    "has_more_results": true,  
    "message": "There is more data for you to enjoy.",  
    "results_count": 500,  
    "total_count": 1381,  
    "position": "c5eedcc23a144b88917adb8a6194d7a4",  
    "next": "https://api.domaintools.com/v1/iris-investigate/?redirect_  
↳ domain=domaintools.com&next=true&position=c5eedcc23a144b88917adb8a6194d7a4",  
    "results": [  
      {  
        ...  
      }  
    ]  
  }  
}
```

The API will stop emitting the `next` field in response to the `next=true` parameter once it reaches the last page of results.

9.4 Using page_size to Limit Page Size

The page size default is 500 and the `page_size` parameter can make the page size smaller. The following example HTTP GET request URL specifies a `page_size` of 100:

```
https://api.domaintools.com/v1/iris-investigate/?redirect_  
↳ domain=domaintools.com&page_size=100
```

9.5 Sorting Results

9.5.1 Using sort_by

Sorting defaults to descending order of `first_seen_since` when no sorting method is specified, as in results for this query:

```
https://api.domaintools.com/v1/iris-investigate/?nameserver_domain=markmonitor.zone
```

Use the `sort_by` parameter to sort by:

- `first_seen_since` (default sort method; defaults to descending)
- `create_date` (default: descending)
- `domain` (default: ascending)
- `risk_score` (default: ascending)

For example, use `sort_by` to sort by `risk_score`:

```
https://api.domaintools.com/v1/iris-investigate/?redirect_
↳ domain=domaintools.com&sort_by=risk_score
```

9.5.2 Using sort_direction

Beginning with a query that sorts a list of domain results by `risk_score` with default ascending order:

```
https://api.domaintools.com/v1/iris-investigate/?redirect_
↳ domain=domaintools.com&sort_by=risk_score
```

Now, with results in descending order with `&sort_direction=dsc`:

```
https://api.domaintools.com/v1/iris-investigate/?redirect_
↳ domain=domaintools.com&sort_by=risk_score&sort_direction=dsc
```

This also works with `create_date` and `first_seen_since`.

9.6 See Also

- [Search Parameters](#) - Learn about filtering to reduce result sets
- [Quota Consumption](#) - Understand how pagination affects quota usage

10 Iris Investigate API quota consumption

Understanding what counts as quota consumption helps you optimize your API usage. The system measures quotas at the group level and resets them each month.

10.1 Duplicate Query Policy for API

Identical API queries within 1 hour do not count against quota. A query is considered identical if it has the same parameters (filters, sorting, pagination) as a previous query made within the last hour.

10.2 Activities That Consume Quota

The following API activities consume your quota:

- Executing a query
- Loading additional pages of results

10.3 Activities That Do Not Consume Quota

The following API activities do not consume your quota:

- Identical queries made within 1 hour of a previous query that counted toward quota

10.4 Additional Information

For complete details on quota consumption policies, including UI-specific policies, search hash reloading, and revisiting search nodes, consult the [Iris API Rate Limits](#) documentation.

10.5 See Also

- [Rate Limits](#) - Complete rate limit documentation
- [Pagination](#) - Understand how pagination affects quota usage

11 Iris Investigate API search

The Iris Investigate API supports powerful search capabilities that enable you to find domains based on various attributes.

11.1 Search Overview

Iris Investigate supports a set of base search parameters and filter parameters. Base search parameters can be used on their own or in combination with each other, while filter parameters refine the base search.

Instead of a domain name, you can provide one or more search fields to the API, such as IP address, SSL hash, email, or more, and Iris Investigate will return any domain name with a record that matches those parameters. This enables “reverse” searching on one or more fields with a single API endpoint.

11.1.1 Example: Search by IP Address

Search for all domains linked to the IP address 199.30.228.112:

```
https://api.domaintools.com/v1/iris-investigate/?ip=199.30.228.112
```

11.1.2 Combining Search Parameters

Queries across multiple parameters are interpreted as a logical AND query, meaning multiple parameters will narrow a search to a smaller result set. The Iris Investigate API does not currently support logical OR queries.

Domain records returned in the result set are identical to records returned from a query for one or more domain names. For example, consider using the [guided pivot counts](#) to surface new ways to expand the result set. Or, you could sort on the risk score (highest to lowest) to show the results to the end user with riskiest domains listed first.

11.2 Search Types

11.2.1 Base Search Parameters

Base search parameters can be used independently or combined with other base parameters and filters. These include searches by:

- Domain attributes (domain name, TLD)
- Infrastructure (IP, nameserver, mail server)
- Identity (email, registrant, registrar)
- SSL/TLS certificates (hash, common name, organization)
- Tracking codes (Google Analytics, AdSense, etc.)
- And more

11.2.2 Filter Parameters

Filter parameters refine base searches by adding constraints such as:

- Date ranges (create date, expiration date, first seen)

- Risk scores
- Geographic location
- Domain status (active/inactive)
- Tags

11.2.3 RDAP and WHOIS Search

Search and filter within parsed WHOIS and RDAP fields, and compare records from both sources.

11.3 Search Examples

11.3.1 Example: Filter by TLD

Search for `domaintools.com` and `domaintools.net` with a filter for a `.com` TLD will surface `domaintools.com` as a result:

```
https://api.domaintools.com/v1/iris-  
↳ investigate/?domain=domaintools.com,domaintools.net&tld=com
```

11.3.2 Example: Search by Email Domain

Search for domains registered with email addresses from a specific domain:

```
https://api.domaintools.com/v1/iris-investigate/?email_domain=example.com
```

11.4 See Also

- [Pagination](#) - Handle large result sets
- [Guided Pivots](#) - Use pivot counts to identify related domains
- [Base Parameters](#) - Complete list of base search parameters
- [Filter Parameters](#) - Complete list of filter parameters

12 Iris Investigate API base search parameters

Base search parameters can be used independently or combined with other base parameters and [filter parameters](#).

12.1 Parameter Reference

Parameter	Description	WHOIS/RDAP Fields Search	Compare To
adsense	Google AdSense tracking code		
baidu_analytics	Baidu analytics code		
contact_name	Contact name from domain registration data	Yes	
contact_phone	Contact phone number from domain registration data	Yes	
contact_street	Contact street address from domain registration data	Yes	
domain	Apex domain		
email	Email address from the most recently available WHOIS record, DNS SOA record or SSL certificate	Yes (for registration emails)	Reverse WHOIS
email_dns_soa	DNS SOA email		
email_domain	Apex domain portion of a WHOIS or DNS SOA email address	Yes (for registration emails)	Reverse WHOIS
facebook	Facebook/Meta tracking code		
google_analytics	Google Analytics tracking code		
ga4	Google Analytics 4 tracking code		

Parameter	Description	WHOIS/RDAP Fields Search	Compare To
google_tag_manager	Google Tag Manager tracking code		
historical_email	Email addresses from historical WHOIS records		
historical_free_text	Free text search of a domain's historical WHOIS records - compare to whois		
historical_registrant	Registrant names from historical WHOIS records		
hotjar	Hotjar tracking code		
iana_id	Registrar IANA code from most recent RDAP record for a domain		
ip	IPv4 address the registered domain was last known to point to during an active DNS check		Reverse IP
mailserver_domain	Only the registered domain portion of the mail server (domaintools.net)		
mailserver_host	Fully-qualified host name of the mail server (mx.domaintools.net)		Reverse MX
mailserver_ip	IP address of the mail server		
matomo	Matomo tracking code		
nameserver_domain	Registered domain portion of the name server (domaintools.net)		Reverse Nameserver
nameserver_host	Fully-qualified host name of the name server (ns1.domaintools.net)		

Parameter	Description	WHOIS/RDAP Fields Search	Compare To
nameserver_ip	IP address of the name server		
redirect_domain	Find domains observed to redirect to another domain name		
registrant	Substring search on the WHOIS registrant field	Yes	Reverse WHOIS
registrant_org	Substring search on the WHOIS registrant org field	Yes	Reverse WHOIS
registrar	Exact match to the WHOIS registrar field	Yes	Reverse WHOIS
search_hash	Encoded search from the Iris Investigate UI		
server_type	Most recent server type retrieved from screenshot gathering		
ssl_alt_names	Query elements/labels from a list of domains/subdomains in certificate Subject Alt Name lists		
ssl_common_name	Certificate Common Name		
ssl_duration	Certificate validity duration, in number of days		
ssl_email	Email address from the SSL certificate		
ssl_hash	SSL certificate SHA-1 hash		
ssl_issuer_common_name	Certificate issuer Common Name		
ssl_org	Exact match to the organization name on the SSL certificate		

Parameter	Description	WHOIS/RDAP Fields Search	Compare To
ssl_subject	Subject field from the SSL certificate		
statcounter_project	Statcounter Project tracker code		
statcounter_security	Statcounter Security tracker code		
tagged_with_all	Domain tagged with all tags		
tagged_with_any	Domains tagged with a specific Iris Tag		
website_title	HTML title from most recent screenshot		
whois	Free text search of a domain's most recent WHOIS record - compare to historical_free_text		
yandex_metrika	Yandex tracker code		

12.2 Usage Examples

12.2.1 Search by IP Address

```
https://api.domaintools.com/v1/iris-investigate/?ip=199.30.228.112
```

12.2.2 Search by Email Domain

```
https://api.domaintools.com/v1/iris-investigate/?email_domain=example.com
```

12.2.3 Search by SSL Hash

```
https://api.domaintools.com/v1/iris-investigate/?ssl_hash=abc123def456
```

12.2.4 Combine Multiple Parameters

Search for domains with a specific IP and nameserver:

```
https://api.domaintools.com/v1/iris-investigate/?ip=199.30.228.112&nameserver_  
↳ domain=example.com
```

12.3 See Also

- [Filter Parameters](#) - Refine your searches with filters
- [Search Overview](#) - Learn about search capabilities
- [RDAP and WHOIS Search](#) - Search within parsed registration data

13 Iris Investigate API filter parameters

Filter parameters refine [base search](#) results by adding constraints. These parameters cannot be used on their own and must be combined with at least one base search parameter.

13.1 Parameter Reference

Parameter	Description	WHOIS/RDAP Fields Search
<code>active</code>	<code>true</code> returns domains that have an entry in the global DNS, OR are listed as registered by the registry. <code>false</code> returns domains that are not in global DNS, AND are not listed as registered by the registry. Exclude filter to return all domains.	
<code>create_date</code>	Only include domains created on a specific date, in YYYY-MM-DD format	Yes
<code>expiration_date</code>	Only include domains expiring on a specific date, in YYYY-MM-DD format	Yes
<code>first_seen_since</code>	Matches domains with a first seen on or after an ISO8601-compliant date/time (including a timezone)	
<code>first_seen_within</code>	Matches domains with a first seen within a specific number of seconds from when the request is made	
<code>ip_country_code</code>	Country code for IP A-record	
<code>not_tagged_with_all</code>	A tag or comma-separated list of tags. Filters for domain that are not tagged with any of the provided tags.	

Parameter	Description	WHOIS/RDAP Fields Search
not_tagged_with_any	A tag or a comma-separated list of tags. Filters for domains that are not tagged with any of the provided tags.	
rank	Popularity rank (replaces Alexa ranking)	
risk_score	Domain Risk Score	
server_type	The web server type	
ssl_not_after	Validity end date for a certificate (YYYY-MM-DD)	
ssl_not_before	Validity begin date for a certificate (YYYY-MM-DD)	
tagged_with_all	A tag or comma-separated list of tags. Filters for domain that are tagged with all of the provided tags.	
tagged_with_any	A tag or a comma-separated list of tags. Filters for domains that are tagged with any of the provided tags.	
tld	Limit results to only include domains in a specific top-level domain (i.e., tld=com, tld=ru)	
website_title	The value of the website's title tag	

13.2 Usage Examples

13.2.1 Filter by TLD

Search for `domaintools.com` and `domaintools.net` with a filter for a `.com` TLD will surface `domaintools.com` as a result:

```
https://api.domaintools.com/v1/iris-
↳ investigate/?domain=domaintools.com,domaintools.net&tld=com
```

13.2.2 Filter by Create Date

Search for domains on a specific IP created on a specific date:

```
https://api.domaintools.com/v1/iris-investigate/?ip=199.30.228.112&create_-  
↪ date=2023-01-15
```

13.2.3 Filter by Risk Score

Search for high-risk domains on a specific nameserver. The `risk_score` parameter accepts an integer threshold and returns domains with a risk score greater than that value:

```
https://api.domaintools.com/v1/iris-investigate/?nameserver_-  
↪ domain=example.com&risk_score=69
```

13.2.4 Filter by Active Status

Search for inactive domains with a specific registrant:

```
https://api.domaintools.com/v1/iris-investigate/?registrant=example&active=false
```

13.2.5 Filter by First Seen

Search for recently discovered domains on a specific IP:

```
https://api.domaintools.com/v1/iris-investigate/?ip=199.30.228.112&first_seen_-  
↪ within=86400
```

13.3 See Also

- [Base Search Parameters](#) - Available base search parameters
- [Search Overview](#) - Learn about search capabilities
- [Work with Tags](#) - Using tag filters effectively

14 Iris Investigate API RDAP and WHOIS search

Learn how to search within parsed WHOIS and RDAP fields, and compare records from both sources.

14.1 Search Within Parsed Fields

Filter by a specific component within a `parsed_whois` or `parsed_domain_rdap` field. For example, filter by Create Date in Parsed WHOIS:

```
https://api.domaintools.com/v1/iris-investigate/?domain=github.com&parsed_whois:create_date=2007-10-09
```

14.2 Compare WHOIS and Domain RDAP Records

By default, queries return standard registration data as part of the response root fields. Registration data is from either Domain RDAP or WHOIS, depending on which record is more complete.

Add a standalone set of parsed fields from the Domain RDAP and/or the WHOIS record to the response with the parameters `parsed_domain_rdap=true` and `&parsed_whois=true`.

14.2.1 Example: Request Parsed Domain RDAP

Extend a query for `domaintools.com` by requesting Parsed Domain RDAP records in addition to the default response:

Query URL

```
https://api.domaintools.com/v1/iris-investigate/?domain=domaintools.com&parsed_domain_rdap=true
```

cURL Example

```
curl -X GET \  
  'https://api.domaintools.com/v1/iris-investigate/?domain=domaintools.com&parsed_whois:create_date=2007-10-09&parsed_domain_rdap=true' \  
  -H 'X-API-Key: YOUR_API_KEY'
```

14.2.2 Response Structure

The API response contains a results array. Each item in the array is a domain object, and within the object for `domaintools.com`, the parsed Domain RDAP record is returned as a nested object under the key `parsed_domain_rdap`.

```
{
```

```
"response": {
  "message": "Enjoy your data.",
  "results_count": 1,
  "total_count": 1,
  "results": [
    {
      "domain": "domaintools.com",
      "popularity_rank": 2622,
      "active": true,
      "create_date": {
        "value": "1998-08-02",
        "count": 871
      },
      "parsed_whois": {
        "registrar": {
          "value": "eNom, LLC",
          "count": 593475
        },
        "registrant_org": {
          "value": "DomainTools, LLC",
          "count": 3
        }
      },
      "parsed_domain_rdap": {
        "registrar": {
          "value": "ENOM, INC.",
          "count": 3448760
        },
        "registrant_contact": {
          "name": {
            "value": "REDACTED REGISTRANT",
            "count": 9948449
          }
        }
      }
    }
  ]
}
```

14.3 Use Cases

14.3.1 Compare Registration Data Sources

Request both `parsed_whois=true` and `parsed_domain_rdap=true` to compare data from both sources:

```
https://api.domaintools.com/v1/iris-investigate/?domain=example.com&parsed_whois=true&parsed_domain_rdap=true
```

14.3.2 Search Historical WHOIS

Use `historical_free_text` to search historical WHOIS records:

```
https://api.domaintools.com/v1/iris-investigate/?historical_free_text=example+corp
```

Compare this to `whois` which searches only the most recent WHOIS record:

```
https://api.domaintools.com/v1/iris-investigate/?whois=example+corp
```

14.4 See Also

- [Base Search Parameters](#) - WHOIS/RDAP searchable parameters
- [Filter Parameters](#) - Date-based filters for registration data
- [Search Overview](#) - General search capabilities

15 Work with domain tags in the Iris Investigate API

Search for domains that have already been tagged within the Iris Investigate UI. All Tags created within your instance of the Iris Investigate UI are accessible via the API.

15.1 Use Domain Tags as a Search Parameter

Use `tagged_with_any` and `tagged_with_all` to perform base searches on tags:

Use Case	Sample Parameter	Expected Result
Retrieve Domains matching one of the specified Tags	<code>?tagged_with_any=watch,monitor</code>	All Domains which have been tagged as 'watch' OR 'monitor'
Retrieve Domains matching a list of Tags	<code>?tagged_with_all=block,dangerous,evil</code>	All Domains which have been tagged with all 3 tags – 'block' and 'dangerous' and 'evil'

15.1.1 Examples

Search for domains tagged with either "watch" or "monitor":

```
https://api.domaintools.com/v1/iris-investigate/?tagged_with_any=watch,monitor
```

Search for domains tagged with all three tags:

```
https://api.domaintools.com/v1/iris-investigate/?tagged_with_all=
↳ all=block,dangerous,evil
```

15.2 Use Domain Tags as a Search Filter

Limit base results by filtering for tagged domains:

Use Case	Sample Parameter	Expected Result
Filter result sets by only displaying Domains matching one of the specified 'Tags'	<code>?nameserver_domain=markmonitor.zone&tagged_with_any=watch,monitor</code>	Only list Domains which are tagged as 'watch' OR 'monitor'
Filter result sets by only displaying Domains matching a List of 'Tags'	<code>?nameserver_domain=markmonitor.zone&tagged_with_all=block,dangerous,evil</code>	Only list Domains which are tagged with all 3 tags – 'watch' and 'dangerous' and 'evil'

Use Case	Sample Parameter	Expected Result
Filter result sets by excluding domains matching one of the specified 'Tags'	?nameserver_ domain=markmonitor.zone¬_tagged_with_any=watch,monitor	Only list Domains which are not tagged with 'watch' or 'monitor'
Filter result sets by excluding domains matching a List of 'Tags'	?nameserver_ domain=markmonitor.zone¬_tagged_with_ all=zerolisted,safe,trusted	Only list Domains which are not tagged as 'zerolisted' and 'safe' and 'trusted'

15.2.1 Examples

Filter nameserver search to only show domains tagged as "watch" or "monitor":

```
https://api.domaintools.com/v1/iris-investigate/?nameserver_  
↳ domain=markmonitor.zone&tagged_with_any=watch,monitor
```

Filter nameserver search to exclude domains tagged as safe:

```
https://api.domaintools.com/v1/iris-investigate/?nameserver_  
↳ domain=markmonitor.zone&not_tagged_with_any=safe,trusted
```

15.3 Tag Parameters Reference

Parameter	Type	Description
tagged_with_any	Base search or filter	Domains tagged with any of the specified tags (OR logic)
tagged_with_all	Base search or filter	Domains tagged with all of the specified tags (AND logic)
not_tagged_with_any	Filter only	Domains not tagged with any of the specified tags
not_tagged_with_all	Filter only	Domains not tagged with all of the specified tags

15.4 See Also

- [Base Search Parameters](#) - Complete list of search parameters
- [Filter Parameters](#) - Complete list of filter parameters
- [Search Overview](#) - Learn about search capabilities

16 Retrieve screenshots with the Iris Investigate API

Retrieve the most recent screenshot for a given domain with Iris Investigate (and other Iris) APIs in a two-step process:

1. [Retrieve the screenshot image URL](#)
2. [Retrieve and optionally resize the image file](#)

Also consult the section on [interpreting screenshot metadata](#), below.

16.1 Step 1: Retrieve the Screenshot URL

Append queries with `screenshots=1` to receive a screenshot JSON object in the response.

16.1.1 Example Queries

The following Iris API queries (with header authentication) include screenshot requests:

- Iris Detect: `https://api.domaintools.com/v1/iris-detect/domains/watched/?screenshots=1`
- Iris Enrich: `https://api.domaintools.com/v1/iris-enrich/?domain=example.com&screenshots=1`
- Iris Investigate: `https://api.domaintools.com/v1/iris-investigate/?domain=example.com&screenshots=1`

16.1.2 Query Parameters

Parameter	Required	Type	Valid Values	Description	Example
<code>screenshots</code>	Yes	flag (1)	1	Triggers screenshot object in response.	<code>screenshots=1</code>

16.1.3 Response Fields

Field	Type	Valid Values	Description	Example
<code>ip_address</code>	string null	IP address	The IP address from which the screenshot was taken.	212.129.31.169
<code>last_attempt</code>	string	ISO 8601 date-time format	The date and time the screenshot was last attempted.	2025-02-17T15:34:55Z
<code>last_seen</code>	string	ISO 8601 date-time format	The date and time the screenshot was last observed	2025-02-17T15:34:55Z

Parameter	Required	Type	Valid Values	Description	Example
crop_height	No	integer	Height in pixels	If the value is less than the height of the image, the image will be cropped vertically to the given crop_height, removing pixels from the bottom of the image.	crop_height=500

16.2.2 Example Query

```
api.domaintools.com/screenshot_image?domain=domaintools.com&ts=2025-04-
↳ 30T14:32:10Z&resize_width=500&crop_height=500&rurl=https://www.domaintools.com/
```

16.3 Interpreting Screenshot Metadata

Use the `last_attempt`, `last_seen`, and `timestamp` to interpret the screenshot.

To confirm that the most recent screenshot was gathered during the current domain lifecycle—and isn't from an older, historic screenshot—compare the screenshot's `timestamp` with the domain's `first_seen` value in the API response. If the `timestamp` is later than `first_seen`, the screenshot was taken during the current domain lifecycle.

16.3.1 Initial Capture

The screenshot was successfully captured during the first and only attempt. Since there have been no subsequent checks, all timestamp fields are identical.

Relationships: `timestamp = last_seen = last_attempt`

Response example:

```
"screenshot": {
  "timestamp": "2025-05-12T14:38:19Z",
  "src": "https://screenshots.ar.domaintools.com/screenshot_im-
↳ age?s3v=1&rurl=http%3A%2F%2Fxi.mi.fr&ts=2025-05-
↳ 12T14%3A38%3A19Z&token=5fbd3aa604682d26d9f1f69810a87bfd75ea25e56eadb3553cef2ba804f28",
  "ip_address": "76.76.21.21",
  "last_attempt": "2025-05-12T14:38:19Z",
  "last_seen": "2025-05-12T14:38:19Z"
}
```

16.3.2 Gather Attempt Was Duplicate of Initial Screenshot

DomainTools re-checked a domain after previously taking an initial screenshot. The result was identical to the existing image and no new screenshot was generated.

Relationships: timestamp < last_seen = last_attempt

- timestamp remains unchanged, preserving the original capture time.
- last_attempt is updated to reflect the most recent screenshot check.
- last_seen is updated to indicate the most recent time the image was confirmed valid.
- Because no new image was needed, last_seen and last_attempt are equal.

Response example:

```
"screenshot": {
  "timestamp": "2025-05-12T10:10:49Z",
  "src": "https://screenshots.ar.domaintools.com/screenshot_image?s3v=1&rurl=http%3A%2F%2Fdomaintools.co.in&ts=2025-05-12T10%3A10%3A49Z&token=28d255010a09d1cc8a3326e1c1406971910293f09ddac0159eb35391f705be57",
  "ip_address": "104.219.250.192",
  "last_attempt": "2025-05-13T17:16:34Z",
  "last_seen": "2025-05-13T17:16:34Z"
}
```

16.3.3 Recapture Failed After Previous Successful Capture

DomainTools successfully captured a screenshot on a previous attempt. The next attempt to capture the screenshot failed — the system was unable to retrieve a new image and could not confirm whether the content had changed.

Relationships: timestamp = last_seen, and last_seen < last_attempt

- timestamp reflects the time the screenshot was originally captured.
- last_seen is equal to timestamp, because that was the last time a screenshot was successfully observed
- last_attempt is more recent than last_seen, but the attempt failed

Response example:

```
"screenshot": {
  "timestamp": "2023-12-19T22:17:10Z",
  "src": "https://screenshots.ar.domaintools.com/screenshot_image?s3v=1&rurl=http%3A%2F%2Ffixhelpdesk.com&ts=2023-12-19T22%3A17%3A10Z&token=8cd6edb822594a0aa485e72f5cbb5414a73b657f7438b770844f46fe886c4ea9",
  "ip_address": "91.195.240.19",
  "last_attempt": "2025-05-13T10:25:04Z",
  "last_seen": "2023-12-19T22:17:10Z"
}
```

16.3.4 Duplicate Capture Followed by Failed Attempt

A screenshot was captured, and subsequently re-captured as a duplicate. A subsequent attempt to refresh the (duplicate) screenshot failed.

Relationships: timestamp < last_seen, and last_seen < last_attempt

- timestamp reflects the original screenshot date
- last_seen is later than timestamp, indicating the image was revalidated as a duplicate on a later date

- `last_attempt` reflects a more recent failed attempt to recapture the screenshot

Example response:

```
"screenshot": {
  "timestamp": "2022-07-02T10:03:43Z",
  "src": "https://screenshots.ar.domaintools.com/screenshot_im-
↳ age?s3v=1&rurl=http%3A%2F%2Fdomaintools.ca&ts=2022-07-
↳ 02T10%3A03%3A43Z&token=6fa06c07ff1b7206ae8f38aadaf4e11b8cf538b7625944e2813a72ebec2c6648",
  "ip_address": "158.85.87.76",
  "last_attempt": "2025-05-13T10:12:04Z",
  "last_seen": "2024-05-22T10:11:09Z"
}
```

16.4 See Also

- [Quick Start](#) - Learn about basic API usage
- [OpenAPI Specification](#) - Complete API specification

17 Monitor newly active domains with the Iris Investigate API

The Iris Investigate API can be used as a powerful monitoring tool to detect newly active domains pointed to certain IPs, hosted on target name servers, redirecting to specific sites, and any other criteria that can be framed in an Iris Investigate API query.

17.1 How It Works

This can be accomplished by adding either `first_seen_since` or `first_seen_within` to the API query. The first seen is the date/time (in UTC) that DomainTools discovers a domain as newly active.

17.2 Implementation Considerations

17.2.1 Basic Monitoring Pattern

Track newly active domains by storing the timestamp of your last query and using it in subsequent requests:

```
# Store last query time
last_check = "2024-02-04T10:00:00Z"

# Query for new domains since last check
query = f"?ip=199.30.228.112&first_seen_since={last_check}"

# Process results...

# Update timestamp for next query
last_check = current_timestamp
```

17.2.2 Choosing Time Parameters

Use the appropriate parameter based on your monitoring frequency:

- **first_seen_within**: Rolling time window in seconds
 - Example: `first_seen_within=3600` for “domains discovered in the last hour”
 - Updates automatically with each query
 - Best for regular polling intervals
- **first_seen_since**: Exact timestamp in ISO8601 format
 - Example: `first_seen_since=2024-02-04T10:00:00Z`
 - Requires storing and updating the timestamp
 - Best when you need to track precise query windows

17.2.3 Handling Large Result Sets

Monitor queries can return large datasets. Plan accordingly:

Pagination: Results exceeding 500 domains require pagination. Check `has_more_results` in the response and use the `position` parameter to retrieve additional pages. Each page retrieval counts against quota.

10,000 Result Limit: Queries matching more than 10,000 domains return error code 413. Narrow your monitoring criteria with additional filters:

```
# Too broad - may exceed 10,000
?nameserver_domain=markmonitor.zone&first_seen_within=86400

# Narrowed with TLD and risk score filters
?nameserver_domain=markmonitor.zone&tld=com&risk_score=69&first_seen_within=86400
```

Consider splitting broad monitoring into multiple queries with different filter combinations.

17.2.4 Quota Optimization

Identical queries within 1 hour do not consume additional quota. For frequent monitoring:

- Query every 15 minutes with `first_seen_within=3600`
- Only the first query each hour counts against quota
- Subsequent identical queries return cached results

This allows frequent checks without excessive quota consumption.

17.3 Use Cases

This capability makes the Iris Investigate API a compelling replacement for DomainTools Enterprise API monitors that currently only return a single domain in their result: Name Server Monitor and IP Monitor.

It also extends monitoring well beyond those endpoints to include monitoring on SSL attributes, tracking codes, registrar and more, with the additional option to narrow to specific TLDs with the `tld` filter.

17.4 Examples

17.4.1 Monitor Domains on a Specific IP

Monitor for newly active domains on IP 199.30.228.112 discovered in the last 24 hours:

```
https://api.domaintools.com/v1/iris-investigate/?ip=199.30.228.112&first_seen_
↪ within=86400
```

17.4.2 Monitor Domains with Specific SSL Certificate

Monitor for newly active domains using a specific SSL certificate hash:

```
https://api.domaintools.com/v1/iris-investigate/?ssl_hash=abc123def456&first_seen_
↪ since=2024-01-01T00:00:00Z
```

17.4.3 Monitor Domains on Nameserver with TLD Filter

Monitor for newly active `.com` domains on a specific nameserver:

```
https://api.domaintools.com/v1/iris-investigate/?nameserver_  
↳ domain=example.com&tld=com&first_seen_within=3600
```

17.5 See Also

- [Filter Parameters](#) - Learn about `first_seen_since` and `first_seen_within`
- [Guided Pivots](#) - Use pivot counts to identify meaningful connections
- [Pagination](#) - Handle large monitoring result sets

18 Integrate with Iris Investigate UI

The Iris Investigate Platform offers a rich UI to search and pivot through domain name data. Nearly all the data in the Iris Investigate “Pivot Engine” pane is accessible in the Iris Investigate API, and the most common research patterns can be easily accomplished with that API.

18.1 When to Use UI Integration

However, even with these capabilities in the API, there remains a number of scenarios where users can only meet their goals with the complete set of tools and resources offered in the Iris Investigate UI. These scenarios include:

- “OR” searches, or nested “AND” and “OR” searches across multiple fields and values
- Ad-hoc investigations into a single domain’s hosting, WHOIS, or screenshot history
- Graph visualization and graph-initiated pivoting
- Passive DNS queries on IPs and hostnames

That means users may start their investigation in the Iris Investigate UI, but then expect to bring the results of their investigation into a third-party product. The `search_hash` parameter in the API makes this possible.

18.2 How It Works

First, a user conducts an investigation in the Iris Investigate UI, potentially building a complex query to find all the connected infrastructure for a given domain name or threat actor. They access the Advanced Search in Iris Investigate and export the encoded representation of their search.

Next, the user provides this encoded string to your solution, and you craft an Iris Investigate API search with that string in the `search_hash` parameter. No other search parameters are required or supported (except the essential authorization parameters that should always be present). The Iris API “unpacks” the encoded search, runs the same query again, and returns the most up-to-date complete result set.

18.3 Using the `search_hash` Parameter

18.3.1 Example Query

```
https://api.domaintools.com/v1/iris-investigate/?search_hash=ENCODED_SEARCH_STRING
```

18.3.2 Example with cURL

```
curl -X GET \  
  'https://api.domaintools.com/v1/iris-investigate/?search_hash=ENCODED_SEARCH_  
↳ STRING' \  
  -H 'X-API-Key: YOUR_API_KEY'
```

18.4 Workflow

1. User conducts investigation in Iris Investigate UI

- Builds complex query with multiple pivots
- Refines results using UI features

2. User exports search hash

- Accesses Advanced Search
- Copies encoded search string

3. Your application receives search hash

- User provides encoded string to your integration
- Your application stores or processes the hash

4. Your application queries the API

- Constructs API call with `search_hash` parameter
- Receives up-to-date results matching the UI search

5. Your application processes results

- Displays, analyzes, or stores the domain data
- Applies additional business logic as needed

18.5 Benefits

- **Leverage UI capabilities:** Users can use advanced UI features not available in the API
- **Seamless integration:** Transfer complex searches from UI to API without reconstruction
- **Always current:** API re-runs the query to provide the most up-to-date results
- **Simplified development:** No need to replicate complex UI search logic in your integration

18.6 See Also

- [Search Overview](#) - Learn about API search capabilities
- [Base Search Parameters](#) - Available search parameters
- [Quick Start](#) - Get started with the API

19 IrisQL: Iris Query Language for API

IrisQL is a text-based query language for the Iris Investigate API that enables complex domain searches with advanced filtering, OR (logical OR) logic, and nested conditions.

19.1 Using IrisQL

Use IrisQL for:

- **Complex OR logic** - Search for domains matching multiple conditions across different fields
- **Programmatic query building** - Dynamically construct queries in your application
- **Reusing UI (user interface) queries** - Export queries from Iris Investigate UI and use them in your API calls
- **Advanced filtering** - Combine multiple operators and conditions in a single request
- **Nested conditions** - Build queries with complex AND (logical AND)/OR combinations

19.2 Getting started

19.2.1 Endpoint

```
POST https://api.domaintools.com/v1/iris-investigate/
```

19.2.2 Authentication

Use Header Authentication by passing your API key in the X-API-Key header. See [Authentication](#) for details on other authentication methods.

19.3 Request format

19.3.1 Basic structure

Send your IrisQL query as the request body:

```
curl -X POST 'https://api.domaintools.com/v1/iris-investigate/' \  
  -H 'X-API-Key: YOUR_API_KEY' \  
  --data-binary @- << IRISQL  
# IrisQL-1.0  
DOMAIN CONTAINS "example"  
AND  
RISK_SCORE GREATER_THAN 70  
IRISQL
```

19.3.2 Version comment

IrisQL queries must begin with the version comment:

```
# IrisQL-1.0
```

This comment identifies the query as IrisQL and specifies the syntax version.

19.3.3 Query parameters

IrisQL works alongside standard query parameters for the Iris Investigate API:

```
curl -X POST 'https://api.domaintools.com/v1/iris-investigate/?page_size=50&sort_
↳ by=risk_score' \
  -H 'X-API-Key: YOUR_API_KEY' \
  --data-binary @- << IRISQL
# IrisQL-1.0
DOMAIN CONTAINS "spam"
AND
RISK_SCORE GREATER_THAN 75
AND
CREATE_DATE WITHIN "The last 30 days"
IRISQL
```

Query parameters:

IrisQL supports standard query parameters for the Iris Investigate API. These parameters control pagination, sorting, formatting, and more. See the [Iris Investigate API pagination guide](#) for the complete list of available parameters.

19.4 Response format

19.4.1 Response structure

IrisQL responses use the identical structure as the Iris Investigate API. See the [Query Limits and Pagination](#) guide for details on response fields like `limit_exceeded`, `has_more_results`, and `position`:

```
{
  "response": {
    "results": [
      {
        "domain": "example.com",
        "risk_score": 85,
        "create_date": "2020-01-15",
        ...
      }
    ],
    "results_count": 50,
    "total_count": 1250,
    "has_more_results": true,
    "limit_exceeded": false,
    "message": "There is more data for you to enjoy.",
    "missing_domains": []
  }
}
```

All domain attributes include pivot counts showing how many domains have each value. For complete response field documentation, see the [Iris Investigate API documentation](#).

19.5 Examples

All IrisQL queries use the POST method to send the query as the request body. Use the X-API-Key header for authentication.

19.5.1 Basic query

Search for domains containing “phishing” or “malware” with a high risk score:

```
curl -X POST 'https://api.domaintools.com/v1/iris-investigate/' \  
  -H 'X-API-Key: YOUR_API_KEY' \  
  --data-binary @- << IRISQL  
# IrisQL-1.0  
DOMAIN CONTAINS "phishing"  
DOMAIN CONTAINS "malware"  
AND  
RISK_SCORE GREATER_THAN 85  
IRISQL
```

19.5.2 Complex query with multiple conditions

Search for high-risk domains created in the last 30 days with specific top-level domains:

```
curl -X POST 'https://api.domaintools.com/v1/iris-investigate/?page_size=100' \  
  -H 'X-API-Key: YOUR_API_KEY' \  
  --data-binary @- << IRISQL  
# IrisQL-1.0  
DOMAIN CONTAINS "bank"  
DOMAIN CONTAINS "finance"  
AND  
CREATE_DATE WITHIN "The last 30 days"  
AND  
TLD IN ["com", "net", "org"]  
AND  
RISK_SCORE GREATER_THAN 75  
IRISQL
```

19.5.3 Query with sorting and pagination

Retrieve results sorted by risk score with pagination:

```
curl -X POST 'https://api.domaintools.com/v1/iris-investigate/?page_size=50&sort_\  
↳ by=risk_score&position=0' \  
  -H 'X-API-Key: YOUR_API_KEY' \  
  --data-binary @- << IRISQL  
# IrisQL-1.0  
DOMAIN CONTAINS "phishing"  
AND  
RISK_SCORE GREATER_THAN 85  
IRISQL
```

19.5.4 Query with XML response format

Request results in XML format:

```
curl -X POST 'https://api.domaintools.com/v1/iris-investigate/?format=xml' \  
  -H 'X-API-Key: YOUR_API_KEY' \  
  --data-binary @- << IRISQL  
# IrisQL-1.0  
DOMAIN MATCHES "example.com"  
IRISQL
```

19.6 IrisQL syntax reference

19.6.1 Query structure

Each IrisQL query consists of:

1. **Version comment** (required): # IrisQL-1.0
2. **Field conditions**: Field name, operator, and value
3. **Logical operators**: AND to combine conditions
4. **Values**: Must be valid JSON primitives - strings in double quotes, numbers without quotes, booleans (true/false), and arrays with square brackets (e.g., ["value1", "value2"])

Syntax notes:

- IrisQL is case-insensitive for field names and operators.
- Extra whitespace between tokens is normalized and doesn't affect query parsing.
- Write each condition on a single line.
- Don't add leading whitespace before conditions.

19.6.2 Field names and API parameters

IrisQL field names correspond to the parameter names used in the Iris Investigate API. These parameter names may differ from the labels shown in the Advanced Search user interface. For example:

- UI label: "Name Server" -> IrisQL field: NAMESERVER_HOST -> API parameter: nameserver_host
- UI label: "Admin Contact Email" -> IrisQL field: ADMIN_CONTACT_EMAIL -> API parameter: admin_contact_email
- UI label: "IP ASN" -> IrisQL field: ASN -> API parameter: asn

For the complete mapping between API parameters and their data sources, see the [Iris Investigate search parameters](#).

19.6.3 Supported fields

Field names use uppercase with underscores. Fields are organized by category for easier reference.

Domain and risk:

Advanced Search Field	IrisQL Field Name
-----------------------	-------------------

Domain Name	DOMAIN
-------------	--------

Advanced Search Field	IrisQL Field Name
Risk Score	RISK_SCORE
First Seen	FIRST_SEEN
TLD	TLD
Create Date	CREATE_DATE
Expiration Date	EXPIRATION_DATE
Active / Status	ACTIVE
Rank / Popularity Rank	RANK

Registration and WHOIS:

Advanced Search Field	IrisQL Field Name
Registrant	REGISTRANT
Registrant Organisation	REGISTRANT_ORG
Registrar	REGISTRAR
WHOIS Record	WHOIS

Contact information:

Advanced Search Field	IrisQL Field Name
Contact Name	CONTACT_NAME
Contact Street	CONTACT_STREET
Contact Phone	CONTACT_PHONE

Email:

Advanced Search Field	IrisQL Field Name
Email	EMAIL
Email Domain	EMAIL_DOMAIN
Email DNS/SOA	EMAIL_DNS_SOA

Email (by contact type):

Contact-specific email fields target particular contact roles in registration data.

Advanced Search Field	IrisQL Field Name
Admin Contact Email	ADMIN_CONTACT_EMAIL

Advanced Search Field	IrisQL Field Name
Billing Contact Email	BILLING_CONTACT_EMAIL
Registrant Contact Email	REGISTRANT_CONTACT_EMAIL
Technical Contact Email	TECHNICAL_CONTACT_EMAIL
WHOIS Email	MISC_EMAIL

Infrastructure:

Advanced Search Field	IrisQL Field Name	Notes
IP	IP	
IP ASN	ASN	
IP Country Code	IP_COUNTRY_CODE	
IP ISP	ISP_NAME	Requires EXACTLY_MATCH for exact value matching
Server Type	SERVER_TYPE	
Name Server	NAMESERVER_HOST	Fully-qualified hostname
Name Server Domain	NAMESERVER_DOMAIN	Registered domain portion only
Name Server IP	NAMESERVER_IP	
Mail Server	MAILSERVER_HOST	Fully-qualified hostname
Mail Server Domain	MAILSERVER_DOMAIN	Registered domain portion only
Mail Server IP	MAILSERVER_IP	

SSL/TLS certificates:

Advanced Search Field	IrisQL Field Name	Notes
SSL Hash	SSL_HASH	SHA-1 hash
SSL Email	SSL_EMAIL	
SSL Subject / SSL Subject Common Name	SSL_SUBJECT	

Advanced Search Field	IrisQL Field Name	Notes
SSL Organization / SSL Subject Organization Name	SSL_ORG	Requires EXACTLY_MATCH for exact value matching
SSL Issuer Common Name	SSL_ISSUER_COMMON_NAME	
SSL Alt Names	SSL_ALT_NAMES	
SSL Duration	SSL_DURATION	Values must be strings
SSL Not After	SSL_NOT_AFTER	
SSL Not Before	SSL_NOT_BEFORE	

Web content:

Advanced Search Field	IrisQL Field Name
Website Title	WEBSITE_TITLE
Redirect Domain	REDIRECT_DOMAIN

19.6.3.1 Web analytics and tracking

All tracker fields use the EXISTS true or EXISTS false pattern to check for presence or absence of tracking codes, or MATCHES "" to search for specific tracking code values.

Advanced Search Field	IrisQL Field Name
Facebook (Meta Pixel)	FACEBOOK
Google Analytics	GOOGLE_ANALYTICS
Google Analytics 4	GA4
AdSense	ADSENSE
Google Tag Manager	GOOGLE_TAG_MANAGER
Hotjar	HOTJAR
Matomo	MATOMO
Baidu Analytics	BAIDU_ANALYTICS
Yandex Metrika	YANDEX_METRICA
Statcounter Project	STATCOUNTER_PROJECT
Statcounter Security	STATCOUNTER_SECURITY

Metadata:

Advanced Search Field	IrisQL Field Name	Notes
Tags	TAGS	Requires array syntax with CONTAINS: TAGS CONTAINS ["value"]

Historical search:

Advanced Search Field	IrisQL Field Name	Notes
Email (Historical)	HISTORICAL_EMAIL	
Registrant (Historical)	HISTORICAL_REGISTRANT	Only supports MATCHES
Free Text (Historical) / WHOIS (Historical)	HISTORICAL_FREE_TEXT	Searches historical WHOIS records

19.6.4 Operators

Advanced Search Operator	IrisQL Operator	Example
Begins With	BEGINS_WITH	DOMAIN BEGINS_WITH "test"
Ends With	ENDS_WITH	DOMAIN ENDS_WITH ".com"
Contains	CONTAINS	DOMAIN CONTAINS "test"
Contains All	CONTAINS_ALL	REGISTRANT CONTAINS_ALL "John Smith"
Doesn't Contain	DOES_NOT_CONTAIN	REGISTRANT DOES_NOT_CONTAIN "privacy"
Doesn't Contain All	DOES_NOT_CONTAIN_ALL	REGISTRANT DOES_NOT_CONTAIN_ALL "privacy protection"
Matches	MATCHES	DOMAIN MATCHES "example.com"

Advanced Search Operator	IrisQL Operator	Example
Doesn't Match	DOES_NOT_MATCH	DOMAIN DOES_NOT_MATCH "test"
In	IN	TLD IN ["com", "net", "org"]
Not In	NOT_IN	TLD NOT_IN ["xyz", "top"]
Exactly In	EXACTLY_IN	DOMAIN EXACTLY_IN ["exam-ple.com", "test.com"]
Not Exactly In	NOT_EXACTLY_IN	DOMAIN NOT_EXACTLY_IN ["spam.com", "malware.com"]
Exists	EXISTS	EMAIL EXISTS true - See Searching by field presence
Greater Than	GREATER_THAN	RISK_SCORE GREATER_THAN 70
Greater Than or Equal To	GREATER_THAN_OR_EQUAL	RISK_SCORE GREATER_THAN_OR_EQUAL 70
Less Than	LESS_THAN or LOWER_THAN	RISK_SCORE LESS_THAN 30
Less Than or Equal To	LESS_THAN_OR_EQUAL_TO or LOWER_THAN_OR_EQUAL	RISK_SCORE LESS_THAN_OR_EQUAL_TO 50
Within	WITHIN	FIRST_SEEN WITHIN "The last 7 months"

IrisQL operators correspond to match operations in Advanced Search. See [Match operations in Advanced Search](#) for details.

19.6.5 Searching by field presence vs. specific values

Some searches need to find records where a field has any value (presence), rather than matching a specific value. IrisQL provides two methods depending on the field type.

Using the EXISTS operator:

The EXISTS operator always requires a boolean value (true or false):

- EXISTS true - Finds records where the field has any value

- EXISTS false - Finds records where the field has no value

```
EMAIL EXISTS true           # Finds records with any email address
EMAIL EXISTS false         # Finds records without an email address
SSL_HASH EXISTS true       # Finds records with any SSL certificate
FACEBOOK EXISTS true       # Finds records with Facebook tracking
GOOGLE_ANALYTICS EXISTS false # Finds records without Google Analytics
```

See the [Web analytics and tracking](#) section for the complete list of tracker fields.

19.6.6 Logical operators

Use logical operators to combine multiple search conditions.

AND operator (logical AND):

The AND operator combines multiple conditions. All conditions must be true for a domain to match. Each condition appears on its own line, with AND on a separate line between them:

```
DOMAIN MATCHES "example.com"
AND
RISK_SCORE GREATER_THAN 70
AND
TLD MATCHES "com"
```

OR operator (logical OR):

Use the logical OR operation in two ways:

Multi-line OR pattern:

When you write multiple conditions for the same field without AND between them, they work as OR logic. This pattern is useful when you need different operators or want to combine OR conditions with other AND conditions:

```
# IrisQL-1.0
DOMAIN CONTAINS "bank"
DOMAIN CONTAINS "finance"
AND
CREATE_DATE WITHIN "The last 1 month"
```

This query finds domains that contain “bank” OR “finance”, AND were created in the last month.

IN operator for OR logic:

For cleaner syntax when matching multiple specific values with the same operator, use the IN operator with *array syntax*:

```
# Instead of multiple lines:
# DOMAIN MATCHES "example.com"
# DOMAIN MATCHES "test.com"
# Use:
DOMAIN IN ["example.com", "test.com"]
```

When to use each approach:

- **Multi-line pattern:** When you need different operators (e.g., CONTAINS and BEGINS_WITH) or complex OR conditions
- **IN operator:** When matching multiple specific values with the same matching logic

19.6.7 Registration data sources

By default, IrisQL searches use registration data, which automatically selects the most appropriate RDAP or WHOIS record. You can also search specifically against RDAP or WHOIS data using prefix syntax.

RDAP-specific search:

```
# IrisQL-1.0
parsed_domain_rdap:ADMIN_CONTACT_EMAIL MATCHES "admin@example.com"
```

WHOIS-specific search:

```
# IrisQL-1.0
parsed_whois:REGISTRANT MATCHES "John Smith"
```

Default behavior (registration data):

```
# IrisQL-1.0
EMAIL MATCHES "admin@example.com"
```

For more information about RDAP (Registration Data Access Protocol) and WHOIS data, see [RDAP and WHOIS data in search](#) in the Iris Investigate User Guide.

19.6.8 Historical search syntax

IrisQL uses the HISTORICAL_ prefix to access historical data:

- HISTORICAL_EMAIL - Search historical email addresses
- HISTORICAL_REGISTRANT - Search historical registrant names (supports MATCHES operator only, not CONTAINS)
- HISTORICAL_FREE_TEXT - Search historical free text

Example:

```
# IrisQL-1.0
HISTORICAL_EMAIL MATCHES "admin@example.com"
AND
RISK_SCORE GREATER_THAN 70
```

Certain fields support historical search. See [Historical search settings](#) in the Iris Investigate User Guide for details.

19.6.9 Data types

IrisQL supports several data types for query values. Understanding these types helps you write correct queries.

Important: JSON format requirement

All IrisQL values must be valid JSON primitives. This means strings must use double quotes (not single quotes), numbers must be valid JSON numbers, and arrays must use proper JSON array syntax with square brackets.

Strings:

Text values always use double quotes:

```
DOMAIN MATCHES "domaintools.com"  
EMAIL MATCHES "admin@example.com"  
REGISTRANT CONTAINS "John Smith"
```

To include a double quote character in a string, escape it with a backslash:

```
REGISTRANT CONTAINS "\"quoted text\""
```

Numbers:

Numeric values don't use quotes:

```
RISK_SCORE GREATER_THAN 90
```

Booleans:

Boolean values (true or false) don't use quotes:

```
ACTIVE MATCHES true
```

For web analytics/tracker fields, see the [Web analytics and tracking](#) section for the EXISTS true/false pattern.

Dates and times:

Relative dates:

Use the WITHIN operator with the pattern: "The last {N} {unit}"

Supported time units: day, week, month, year (singular or plural)

```
FIRST_SEEN WITHIN "The last 7 months"  
FIRST_SEEN WITHIN "The last 1 day"  
CREATE_DATE WITHIN "The last 30 days"  
EXPIRATION_DATE WITHIN "The last 1 year"
```

Absolute dates:

Use comparison operators with ISO 8601 date format (YYYY-MM-DD):

```
FIRST_SEEN GREATER_THAN "2024-01-01"  
CREATE_DATE LESS_THAN "2025-01-01"  
EXPIRATION_DATE GREATER_THAN_OR_EQUAL "2026-01-01"
```

Note on timezones: The system interprets dates without times as midnight Coordinated Universal Time (UTC). The UI displays dates in your local timezone, which may show a different date. For example, "2024-01-01" (midnight UTC) displays as 2023-12-31 in US Eastern timezone (UTC-5).

IP addresses:

IP addresses are represented as quoted strings:

```
IP MATCHES "192.168.1.1"  
IP GREATER_THAN "192.168.1.0"  
IP LESS_THAN "192.168.2.0"
```

For Classless Inter-Domain Routing (CIDR) ranges, use the IN operator with array syntax:

```
IP IN ["192.168.1.0/24"]
```

Arrays:

Some operators and fields require array values. Arrays use square brackets with comma-separated values (JSON array format):

IN operator:

Always requires an array:

```
TLD IN ["com", "net", "org"]  
DOMAIN IN ["example.com", "test.com"]
```

CONTAINS operator with TAGS field:

Requires an array:

```
TAGS CONTAINS ["malicious"]  
TAGS CONTAINS ["malicious", "phishing"]
```

Array syntax rules:

- Use square brackets: [...]
- Quote string values: ["value1", "value2"]
- Separate values with commas
- No quotes around numbers in arrays (if applicable)

19.7 Rate limits

IrisQL requests are subject to the same rate limits as the Iris Investigate API. See [Rate Limits](#) for details.

19.8 See also

- [Iris Investigate API documentation](#)
- [OpenAPI Specification](#)
- [IrisQL in Iris Investigate UI](#)
- [Authentication](#)
- [Error Codes](#)