API Documentation (public version)

This document describes the public API of Intelligence X to perform searches based on selectors and retrieve the results. For more details about Intelligence X please visit https://intelx.io/.

For any questions please contact info@intelx.io.

Version 1 from 12/5/2018. History of this document:

VERSION	DATE	NOTE
1	12/5/2018	Initial version.
2	4/5/2019	Added information about the SDK.

Table of Contents

Basics

The API only uses HTTP GET and POST requests. Payload data is usually JSON encoded. Every request must be authenticated by including an API key in the request. All times are UTC.

This document prerequisites an API key and API URL which you receive separately. Any limitations are stated in your license document.

Authentication

Every request must be authenticated. The API key is an UUID, for example "9dfa1da9-df94-4ecc-99ed-cf828a12a1fc".

The preferred method is sending the API key in the "x-key" header.

Alternatively, you can place the API key as parameter in the URL:

Format:	&k=[key]
Example:	&k=9df61df0-84f7-4dc7-b34c-8ccfb8646ace

The API key is tied to your organization. Please make sure to not accidentally leak it to the public.

When using the Intelligence X API as part of your online application or service, make sure to make all requests only server side (for example via PHP or Node.JS) and not client-side via JavaScript/Ajax.

Limitations and Timeouts

Unless otherwise agreed in your license, please do not hit the API with more than 1 request/second.

Depending on your license agreement your API key may be restricted to certain functions and may have daily limits on those functions. The API will return HTTP 401 Unauthorized in case your key does not have access to the function.

Daily limits are implemented as "credits per day per function". Whether there are daily limits depends on your license. Credits get reset at midnight UTC. In case all credits are used, the API returns HTTP 402 Payment Required.

Different API instances may have different technical limitations and timeouts. Please refer to your license document for the details.

Buckets

A bucket is a data category such as "darknet.tor". Depending on your license your API key may have access only to certain buckets. If your license includes access to a bucket (as example "web"), it means you automatically have access to all buckets that are starting with it (for example "web.public.kp").

Buckets contain "items", which are the individual results. Each item has a unique identifier called the "System ID" which is an UUID. Individual results are identified by their System ID. For example, a single historical website copy is an item and has its own System ID. A different historical copy of the same website will have a different System ID.

Legal

The Intelligence X data, API and use of the service is covered by the Terms of Service published at https://intelx.io/terms-of-service as well as the Privacy Policy at https://intelx.io/privacy-policy.

HTTP Responses

The API may return these HTTP status codes:

HTTP Status	Info
200 OK	Success, with data
204 No Content	Download with content-disposition but item not available
400 Bad Request	Invalid input. Returned if the encoding is invalid or a required parameter is missing.
401 Unauthorized	Authenticate: Access not authorized. This may be due missing permission for API call or to selected buckets.
402 Payment Required	Authenticate: No credits available.
404 Not Found	Item or identifier not found

intelx.io

If you plan to integrate the API into your website, product or service, you can link the full result to view on https://intelx.io. The format is “https://intelx.io/?did=[System ID]”.

Example full link: https://intelx.io/?did=7597be0c-589c-4e1d-b300-92bd577b4e47

API Key

You can find your API key on intelx.io in the developer tab https://intelx.io/account?tab=developer (requires to be signed in).

Alternatively, to test your scripts you may use the public API key:

API URL	https://public.intelx.io/	
API Key	9df61df0-84f7-4dc7-b34c-8ccfb8646ace	
Access to buckets	pastes	Preview
	darknet.tor	Preview
	darknet.i2p	Preview
	web.public.kp	Full

This public API key is free of charge; however, it has certain limitations such as:

- Searches: 50 /day
- Phonebook lookups: 20 /day
- Max. results per search: 40
- It will only return the first 1.000 characters of items data and redact everything else

These limitations may change at any time. The public API key may not be used in enterprise production environments (contact info@intelx.io for the available enterprise plans).

SDK

The software development kit (SDK) is published at https://github.com/IntelligenceX/SDK.

Overview

The functions are grouped into these categories:

1. Search for results based on selectors and retrieving the results:

`/intelligent/search`	Submits an intelligent search request
`/intelligent/search/result`	Returns selected results
`/intelligent/search/terminate`	Terminates a search

2. Phonebook search. This allows to query selectors, for example all emails known for a particular domain.

`/phonebook/search`	Submits a phone book alike search
`/phonebook/search/result`	Returns results

3. Reading items data or previewing it. A preview shows max. the first 1000 characters.

`/file/read`	Reads an items data for download
`/file/view`	Reads an items data for detailed inline view
`/file/preview`	Reads an items data for preview

Some functions ending with "/human" just differ in their non-human versions that they include fields that are translating certain fields to human speech.

Search Functions

The search functions allow to perform searches based on selectors.

A typical function that queries search results from Intelligence X would look like this:

1. Submit search request to /intelligent/search and retrieve the search ID.
2. Wait for results to be sorted and ready, at least about ~400ms
3. Request the results from /intelligent/search/result with the search ID. Request them until the response is no longer 0 Success with results or 3 No results available, but keep trying.
4. Request the preview via /file/preview or the full data via /file/read for the search results as required.

If you no longer need results and the result function did not indicate that the search is finished, you should manually terminate it by calling /intelligent/search/terminate in order to free system resources.

/intelligent/search

This submits a search request for a selector.

Request	POST	/intelligent/search with JSON IntelligentSearchRequest
Response	200	Success with JSON IntelligentSearchRequestResponse
	400	Invalid JSON

Example request below. The example terminates 2 previous searches at once and searches for "test.com" sorted by relevancy with max 1000 results.

```
{
    "term": "test.com",
    "buckets": [],
    "lookuplevel": 0,
    "maxresults": 1000,
    "timeout": 0,
    "datefrom": "",
    "dateto": "",
    "sort": 2,
    "media": 0,
    "terminate": ["d57897e8-9333-4ed9-8a00-55bcdcf97fab", "61202067-543e-4e6a-8c23-11f9b8f0
08cf"]
}
```

Example response:

```
{
    "id": "61202067-543e-4e6a-8c23-11f9b8f008cf",
    "softselectorwarning": false,
    "status": 0
}
```

Explanation of the IntelligentSearchRequest fields:

- By using the terminate field multiple (or none) previous searches can be terminated when starting a new one. This terminates normal search requests and Phonebook lookups when their IDs are supplied. If a search is no longer needed, it should be terminated via /intelligent/search/terminate to free system resources.
- The sort parameter may be:

SORT	INFO
0	No sorting
1	X-Score ASC. This equals to "least relevant items first".
2	X-Score DESC. This equals to "most relevant items first".
3	Date ASC. This equals to "oldest items first".
4	Date DESC. This equals to "newest items first".

- If the date filters are used both from and to date must be supplied.
- The timeout is in seconds, if 0 it means default.
- The media parameter can be one of the following (0 means not used as filter):

MEDIA	INFO
0	Not set
1	Paste Document
2	Paste User
3	Forum
4	Forum Board
5	Forum Thread

6	Forum Post
7	Forum User
8	Screenshot of a Website
9	HTML copy of a Website
10	Invalid, do not use
11	Invalid, do not use
12	Invalid, do not use
13	Tweet
14	URL, high-level item having HTML copies as linked sub-items
15	PDF document
16	Word document
17	Excel document
18	Powerpoint document
19	Picture
20	Audio file
21	Video file
22	Container files including ZIP, RAR, TAR and others
23	HTML file
24	Text file

- The term must be a strong selector. These selector types are currently supported:
    - Email address
    - Domain, including wildcards like *.example.com
    - URL
    - IPv4 and IPv6
    - CIDRv4 and CIDRv6
    - Phone Number
    - Bitcoin address
    - MAC address
    - IPFS Hash
    - UUID
    - Storage ID
    - System ID
    - Simhash
    - Credit card number
    - IBAN

Soft selectors (generic terms) are not supported! If you supply a generic term the response will have the field “softselectorwarning” set to true.

- Buckets is a list of buckets to search in. If none is specified, the system searches in all buckets that available to the API key. These are the available buckets currently:

Bucket	License	Info
pastes	PRO	Pastes from various pastebin sites
darknet.tor	PRO	Onion sites from Tor
darknet.i2p	PRO	Eepsites (.i2p) from I2P
web.public.kp	Free	North Korean public websites

- The maxresults field tells how many results to query maximum per bucket. The total amount of all returned results might be therefore bigger than the supplied limit.
- The lookuplevel field should always be 0.

Explanation of the response IntelligentSearchRequestResponse fields:

- The field id returns the search ID that can be used to query the search results and terminate the search.
- Status is one of the following:

SORT	INFO
0	Success, search ID is valid
1	Invalid term
2	Error, max concurrent searches per API key

If the API key does not have access to the specified bucket, it will return HTTP 401 Not Authorized.

If there is a credit limit per day and all credits are used, the API returns with HTTP 402 Payment Required.

/intelligent/search/result

This returns the actual results, called items, of a search. The default limit (if none set) is 100. Human text fields are HTML escaped.

Request	GET	/intelligent/search/result?id=[UUID]&offset=[optional offset]&limit=[max records]&previewlines=[optional lines]
Response	200	Success with JSON IntelligentSearchResult containing the fields status and records.
	400	Invalid Search ID

The returned IntelligentSearchResult structure contains a status code indicating:

STATUS	INFO
0	Success with results
1	No future results available, stop trying. All results were delivered and the search is terminated. Note: Records may be returned with this status code.
2	Search ID not found
3	No results yet available, but keep trying.

Example response with status code 1 indicating no more results available:


```
{
    "records": [],
    "status": 1
}
```


Example response with data:


```
{
    "records": [{
        "systemid": "f13fdfc7-2ef6-42a4-bccf-61ff57971bd9",
        "storageid": "143cd96f978d7a7d43127b65b096c99462354098db599cf0daa684a8fb69932b7f0b0
420973bc99a83193ff98970a89a5b2e09bb0b4957e13427b2c0394d3407",
        "instore": true,
        "size": 4315485,
        "accesslevel": 0,
        "type": 1,
        "media": 1,
        "added": "2017-12-14T07:35:50.998926Z",
        "date": "2017-12-14T07:35:05Z",
        "name": "",
        "description": "",
        "xscore": 68,
        "simhash": 15623487490024384908,
        "bucket": "pastes",
        "keyvalues": null,
        "tags": null,
        "relations": [{
            "target": "07c11339-1066-4906-9490-9b93d2c35690",
            "relation": 0
        }],
        "accesslevelh": "Public",
        "mediah": "Paste",
        "simhashh": "d8d1c7186ba9a98c",
        "typeh": "Text",
        "tagsh": null,
        "friends": [{
            "systemid": "f13fdfc7-2ef6-42a4-bccf-61ff57971bd9",
            "date": "2017-12-14T07:35:05Z",
            "name": "Paste ",
```

```
            "inline": {
                "systemid": "f13fdfc7-2ef6-42a4-bccf-61ff57971bd9",
                "storageid": "143cd96f978d7a7d43127b65b096c99462354098db599cf0daa684a8fb699
32b7f0b0420973bc99a83193ff98970a89a5b2e09bb0b4957e13427b2c0394d3407",
                "instore": true,
                "size": 4315485,
                "accesslevel": 0,
                "type": 1,
                "media": 1,
                "added": "2017-12-14T07:35:50.998926Z",
                "date": "2017-12-14T07:35:05Z",
                "name": "",
                "description": "",
                "xscore": 68,
                "simhash": 15623487490024384908,
                "bucket": "pastes",
                "keyvalues": null,
                "tags": null,
                "relations": [{
                    "target": "07c11339-1066-4906-9490-9b93d2c35690",
                    "relation": 0
                }],
                "accesslevelh": "Public",
                "mediah": "Paste",
                "simhashh": "d8d1c7186ba9a98c",
                "typeh": "Text",
                "tagsh": null
            }
        }, {
            "systemid": "07c11339-1066-4906-9490-9b93d2c35690",
            "date": "2017-11-21T03:27:42Z",
            "name": "Paste User monkeypoo",
            "inline": {
                "systemid": "07c11339-1066-4906-9490-9b93d2c35690",
                "storageid": "",
                "instore": false,
                "size": 0,
                "accesslevel": 0,
                "type": 1001,
                "media": 2,
                "added": "2017-11-21T03:27:55.040291Z",
                "date": "2017-11-21T03:27:42Z",
                "name": "monkeypoo",
                "description": "",
                "xscore": 51,
                "simhash": 0,
                "bucket": "pastes",
                "keyvalues": null,
                "tags": null,
                "relations": null,
                "accesslevelh": "Public",
                "mediah": "Paste User",
                "simhashh": "0000000000000000",
                "typeh": "User",
                "tagsh": null
            }
        }],
        "randomid": "f2737370-e182-4707-ad7a-a8f6a3320934",
    }],
    "status": 0
}
```


Friends are items that are linked. For example, when the search result is a historical Website copy, all other historical copies of the same website may be returned as friends.

The offset parameter can be used to get items from a specific place in the item queue. It should not be used normally.

The limit parameter limits how many items are returned with the call.

The previewlines parameter can be used when later calls to /file/preview are intended in order to use the HTTP/2 Push mechanism.

The response structure IntelligentSearchResult contains a field records which is an array of items (the actual results). An item has many fields:

- systemid: The unique identifier of the item.
- storageid: The storage id used to read the actual data of the item.
- instore: Whether the item has actual data and the storageid is valid.
- size: Size in bytes of the actual data
- accesslevel: 4 Indicates that only the preview is available with the API key.
- type is the low-level data type:

TYPE	INFO
0	Binary/unspecified
1	Plain text
2	Picture
3	Video
4	Audio
5	Document file including office documents and PDF
6	Executable file
7	Container
1001	User
1002	Leak
1004	URL
1005	Forum

- media is the high-level data type:

MEDIA	INFO
0	Invalid
1	Paste Document
2	Paste User
3	Forum
4	Forum Board
5	Forum Thread
6	Forum Post
7	Forum User
8	Screenshot of a Website
9	HTML copy of a Website
10	Invalid, do not use
11	Invalid, do not use
12	Invalid, do not use
13	Tweet
14	URL, high-level item having HTML copies as linked sub-items
15	PDF document
16	Word document
17	Excel document

18	Powerpoint document
19	Picture
20	Audio file
21	Video file
22	Container files including ZIP, RAR, TAR and others
23	HTML file
24	Text file

- added: When the item was indexed
- date: The date of original record if available, otherwise set to the same as added.
- name: Name or title
- description: Typically, not used.
- xscore: Relevancy score, between 0-100.
- simhash: Similarity hash, depending on the content type.
- bucket: The bucket in which the result was found
- keyvalues: Not used.
- tags: Additional information on the item
- relations: Identifiers of related items.

/intelligent/search/terminate

To terminate a search, use this function. If a search is already terminated, nothing happens.

Request	GET	/intelligent/search/terminate?id=[UUID]
Response	200	Empty

Phonebook Functions

/phonebook/search

To submit a Phone Book alike search, use this function.

Request	POST	/phonebook/search with JSON IntelligentSearchRequest
Response	200	Success with JSON IntelligentSearchRequestResponse
	400	Invalid Input

For documentation of the IntelligentSearchRequest structure see /intelligent/search. This call can terminate previous normal and phonebook lookups by using the terminate parameter.

/phonebook/search/result

To return Phone Book results use this function. The default limit (if none set) is 100. Human text fields are HTML escaped.

Request	GET	/phonebook/search/result?id=[UUID]&offset=[optional offset]&limit=[max records]
Response	200	Success with JSON PhoneBookSearchResult. Check the field status.

The PhoneBookSearchResult structure matches the status codes returned in IntelligentSearchResult:

STATUS	INFO
0	Success with results
1	No future results available, stop trying. All results were delivered and the search is terminated. Note: Records may be returned with this status code.
2	Search ID not found
3	No results yet available, but keep trying.

File Functions

/file/read

To read an items data use this API. It can be used for direct data download.

For Type 1 the "Content-Type" header will be set to "application/octet-stream" and also the "Content-Disposition" header will be set to a filename. The filename is, unless overwritten by the &name= parameter, defined based on the provided identifier together with an extension that matches the items type. If the System ID is provided and the item indicates text as media type, it will sanitize the line endings to 'CR LF'.

Either the Storage ID or the items System ID must be specified. For Type 0 (raw binary) it is better to use the Storage ID because it is faster; if the System ID is specified, an item lookup is required.

The max download size is limited by the config setting FileReadMax.

Request	GET	/file/read?type=[download type]&storageid=[storage identifier]&systemid=[system identifier]&bucket=[optional bucket]
Response	200	Data as payload according to the type
	400	Invalid Input
	404	Item not found
	204	Item unavailable and downloading done with Content-Disposition header set (type 1).

TYPE	INFO
0	Raw binary
1	Raw binary with content disposition and optional file &name=[name] as parameter Filename: "[System ID].bin" or "[Storage ID].bin" unless specified in the name parameter

/file/view

This API is similar to `/file/read`, its documentation applies here. The returned data may be sanitzied for showing it to the end-user. The output is limited by the config setting `FileViewMax`.

For text the default behavior (unless overwritten by `&escape=0`) is to escape HTML characters for safe placement of the text in HTML.

Request	GET	/file/view?f=[format]&storageid=[storage identifier]&bucket=[optional bucket]
Response	200	Data as payload according to the type
	400	Invalid Input
	404	Item not found

FORMAT	INFO
0	Text view, any non-printable characters shall be removed, UTF-8 encoding.

1	Hex view of data.
2	Auto-detect hex view or text view.
3	Picture view.
4	Not supported.
5	HTML inline view. Content will be sanitized and modified!
6	Text view of PDF. Content will be automatically converted.
7	Text view of HTML.
8	Text view of Word files (DOC/DOCX/RTF).

/file/preview

This reads an items data for preview. The supported preview output formats are:

1. Text: In case the item media is HTML, PDF or Word it will convert it to text. The returned text is in any case limited to 1000 characters max. The default returned max lines is 12 but can be specified as optional parameter `&l=[Max Line Count]`. The returned text is HTML encoded if not disabled via `&e=0`.

2. Picture: If the media is picture, it will scale it down and convert it to a JPG image.

Request	GET	/file/preview?c=[Content Type]&m=[Media Type]&f=[Target Format]&sid=[Storage Identifier]&b=[Bucket]&e=[0|1]
Response	200	Data as payload according to the type
	400	Invalid Input

The Content Type, Media Type, Storage ID and Bucket parameters must be the same as the source item.

The target format must be 0 for text or 1 for picture (only valid for Media=Picture). For text output there is the optional `&e=` parameter enabling HTML escaping of the text (default if not specified is true).

In case there are problems with reading, processing or encoding the data a generic text or picture will be returned informing the user about the problem.