

Categorization SDK

The document contains information about Categorization SDK and describes integration and interaction with it

- [Configuration & Integration](#)
- [Categories](#)
- [Canonical form of a domain \(or URL\)](#)
- [Local database](#)
- [Example](#)
- [Error Information](#)

Configuration & Integration

Introduction

This documentation contains the following solutions:

- The ability to find the categories of a given URL by searching the local database located on the user's terminal device (PC, router, embedded device). In this case, an Internet connection is not required and no network requests are made. This type of database is called the local database.
- The ability to find categories of a given URL using HTTPS access to the SafeDNS cloud. To find a category of domains an internet connection is required and a username/password is required as well to get access to the library. But there is no need to have a local database on the user's computer.
- The ability to save URL categories received from the SafeDNS cloud in the local disk cache for faster access to them later. The data saved in the disk cache is not deleted after restarting the computer and restarting the application using this cache.
- The ability to cache the received categories of a URL in the computer's RAM to speed up their subsequent search. The cache sizes can be set during the library assembling.
- The ability to receive and install regular updates of the local database categories from the SafeDNS cloud. Downloading the entire database is available: all categories ~ 1.6GB; smaller custom builds are also possible.
- The performance of the library on average hardware is approximately 60k requests per second.

The solution is implemented in the programming language "C". It's provided with the user interface in the form of functions of the "C" programming language. The solution can also be integrated into a Python project.

SDK integration into the final product

The SDK is easy enough to integrate into the final software product developed in the C or C++ programming language. It is also possible to integrate it into the final Python solution.

To use the library, you need to get the library sources from repository [here](#), configure the library, build and install it.

Configuring the url2cat library

Necessary utilities to assemble and configure the library:

- **cmake** (version 3.15 or higher)
- **make** (version 4.2 or higher)
- **gcc**
- **sphinx** (version 1.8.5 or higher) (module sphinx_rtd_theme)

Necessary libraries (Ubuntu 18 or higher):

- **openssl-1.1.1s (or 3.0.6)**
- **libssl-dev**
- **pkg-config**
- **libssl-dev**

Configuration

1. Go to the library source directory
2. Configure the library using the command

```
cmake -S . -B build -DCMAKE_BUILD_TYPE=Release -DURL2CAT_SERVER=yes -DURL2CAT_DATABASE=yes -
DURL2CAT_LIBRARY=static -DURL2CAT_LOCALE=en
```

Or edit according to your needs and run the bash script located in `liburl2cat` directory - `create_build.sh`.

To run the script perform:

```
sudo chmod +x create_build.sh
./create_build.sh
```

Using the following cmake commands you can configure library functions:

- `URL2CAT_SERVER` - request to the SafeDNS server (yes, no)
- `URL2CAT_DATABASE` - request to the local database (yes, no)
- `URL2CAT_LIBRARY` - library assembling type (static & shared)
- `URL2CAT_MAX_NUMBER_CATEGORY` - number of defined categories (Default number: 5)
- `URL2CAT_HASH_TYPE` - Hash type (MD5, SHA256, SHA512. Default type - MD5)
- `URL2CAT_HASH_LEN` - Hash length (full - hash is not truncated before searching in the database. Half extract the first 8 bytes of the hash. Default length - half)
- `URL2CAT_CACHE` - category entry cache size (dynamic / static)

3. Build the project using the command:

```
cmake --build build
```

4. Copy the following library files to your project:

- `build_release/lib/liburl2cat.a` or `build_release/lib/liburl2cat.so`
 - `include/url2cat.h`
-

Using the library in the project:

Before using the library, needs to initialize it using the `s_url2cat_setting` structure. The structure has the following fields:

- `cache_size` - cache size in bytes (if set to 0 the cache is not used)
- `db_name` - the name of the database
- `db_download_scheme` - protocol for updating the database
- `db_download_host` - database update host
- `db_download_port` - database update port
- `db_download_path` - database update path
- `db_download_user` - login to updating the database
- `db_download_password` - password to updating the database
- `server_scheme` - protocol for connecting to the SafeDNS server
- `server_host` - host of connection to the SafeDNS server
- `server_port` - port for connecting to the SafeDNS server
- `server_path` - path to connect to the SafeDNS server
- `server_user` - login to connect to the SafeDNS server
- `server_password` - password to connecting to the SafeDNS server

For initialization use the function - `url2cat_init(s_url2cat_setting * setting)`

To get a category use the function:

```
url2cat_category(char * url, size_t len_url, s_url2cat_category ** category, size_t * number_category)
```

where the structure `s_url2cat_category` has the following fields:

- `type` - the number of category
- `type_name` - the name of category

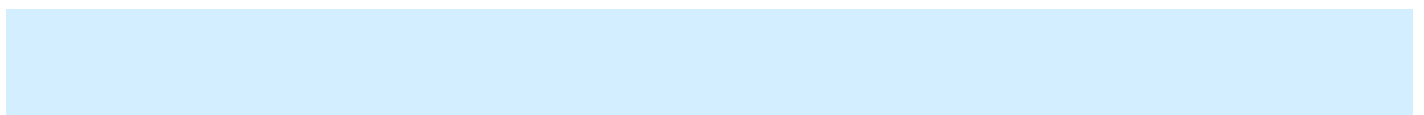
After finishing using the library you will need to deinitialize it with the command: `url2cat_deinit()`

The database can be updated while the library is running using the command:

```
url2cat_database_update(s_url2cat_setting * setting)
```

A request for recategorization can be sent while the library is running using the command:

```
url2cat_recategorize(char * url, size_t url_size, char * category_name, size_t category_name_size)
```



Examples of integration the solution into simple projects running on "C" can be found in the [/example](#) directory.

Categories

ID	Category	Description
3	Malware	Sites spreading malware
4	Phishing & Typosquatting	Sites deceiving internet users (e.g. fake pages, scams, fraud)
5	Online Ads	Advertising systems and banner networks
6	Drugs	Sites advertising or selling drugs
7	Tasteless	Sites containing excessive amounts of tasteless language or unmoderated forums
8	Academic Fraud	Sites with plagiarism, academic fraud, etc.
9	Parked Domains	Sites without any content that are temporarily placed at the domain registrar; are often used for virus propagation
10	Hate & Discrimination	Sites spreading the propaganda of aggression, racism, terrorism
11	Proxies & Anonymizers	Sites containing ways to bypass content filters
12	Botnets & C2C	Malicious websites
13	Adult Sites	Adult content, mostly about sex but without pornography
14	Alcohol & Tobacco	Sites selling or containing information about alcohol and tobacco
15	Dating	Dating websites
16	Pornography & Sexuality	Sites containing pornography in any form
17	Astrology	Occult and astrology sites
18	Gambling	Casinos, lotteries, and other gambling systems

ID	Category	Description
19	Child Sexual Abuse (IWF)	Sites containing child sexual abuse images, criminally obscene adult and child sexual abuse content from a list compiled by Internet Watch Foundation/IWF (UK)
20	Torrents & P2P	Torrent trackers and peer-to-peer networks
21	File Storage	File-sharing sites, file archives
22	Movies & Video	Online movies and videos
23	Music & Radio	Internet radio and music archives
24	Photo Sharing	Archives of photos, photo galleries
26	Chats & Messengers	Online chats and instant messengers, messaging systems servers
27	Forums	Web forums
28	Games	Computer and video games websites
29	Social Networks	Social networking services (e.g. Facebook)
30	Entertainment	Entertainment portals; cafes and restaurants; sites with information about leisure
31	German Youth Protection	Sites considered unsafe by Federal Department for media harmful to young persons (BPjM, Germany)
32	Automobile	Sites about cars and other types of transport
33	Blogs	Systems of mass hosting, personal websites, blogs
34	Corporate Sites	Sites of commercial organizations
35	E-commerce	Online shops
36	Education	Websites of educational institutions, educational portals
37	Finances	Sites of banks and other financial institutions
38	Government	Official government sites
39	Health & Fitness	Sites of hospitals, medical centers, and health care organizations; health portals
40	Humor	Humorous and entertaining sites, often tasteless

ID	Category	Description
41	Jobs & Career	Job portals
42	Weapons	Websites about weapons and army
43	Politics, Society, and Law	Political news; political parties and organizations; legal organizations; laws
44	News & Media	News agencies, media sites
45	Non-profit	Sites of non-profit organizations
46	Portals	Common portals
47	Religious	Sites of religious and anti-religious organizations
48	Search Engines	Google, Bing, DuckDuckGo, etc.
49	Computers & Internet	Sites about IT, software, internet, and computers
50	Sports	Sites about sports or sports organizations
51	Science & Technology	Scientific organizations sites; science news
52	Travel	Sites about tourism and travel
53	Home & Family	Sites about home, family, and hobbies
54	Shopping	Sites with shopping unrelated to online stores
55	Arts	Museums, art portals, and galleries
56	Webmail	Webmail systems
57	Real Estate	Sites with information about sales and purchases of the real estates
58	Classifieds	Classifieds
59	Business	Sites about businesses and economics
60	Kids	Sites for children

ID	Category	Description
63	Trackers & Analytics	Web analytics systems, including user tracking
65	Child Sexual Abuse (Arachnid)	Sites containing child sexual abuse material from a list compiled by the Canadian Centre for Child Protection (CAN) as part of its Project Arachnid
66	Cryptojacking	Sites illegally mining cryptocurrencies
67	Online Libraries	Sites of online libraries
70	DGA	DGAs are algorithms detected in various malware families, which are used to periodically generate a large number of domain names that can be used as a default with their management and control servers.
71	Ransomware	Sites spreading ransomware
72	Generative AI	Sites of well-known AI services, chatbots, and text/picture generators
100	Contentless Domains	These sites may lack any meaningful or legitimate content, making their purpose unclear or unreliable and due to their dubious nature, they pose potential risks to users

Canonical form of a domain (or URL)

URLs of resources are stored in the database in their canonical form, which means that before requesting a list of categories, the URL needs to be canonicalized. In other words, you need to get the canonical representation of the URL.

A single URL resource pointer consists of several parts, but we are only interested in two of them:

- Domain name (`domain`)
- Path (`path`)

Here is a random URL as an example:

```
directory.google.com/example/test.php?key=value&one=1
```

Where the domain is:

```
directory.google.com
```

and this is the URL path:

```
/example/test.php?key=value&one=1
```

Since the URL points to a resource, it can be written in various forms and in a variety of ways. For a direct search in the database, it is necessary to bring all the various forms of URLs pointing to one resource to one canonical form. This is very important for getting the correct categorization of the requested URL.

We use the standard version of URL canonicalization from the Google Safe Browsing project

<https://developers.google.com/safe-browsing/> with the API version higher 2.

This project describes techniques for resource identifier canonicalization, examples and algorithms can be found on the page:

<https://developers.google.com/safe-browsing/v4/urls-hashing#canonicalization>

Examples of canonicalization

Source URL	Canonical URL
<code>http://evil.com/foo-bar-baz</code>	<code>http://evil.com/foo</code>

http://host/%25%32%35	http://host/%25
http://evil.com/foo-bar-baz	http://evil.com/foo
http://test.com/path/../	http://test.com/

Canonicalization in the context of the url2cat library

As described above, you can get a canonical form of a URL. But due to the nature of URLs, you cannot rely on one URL. To find the best match for it in the database, you need to generate derived URLs by alternately discarding parts of the primary URL from the left and right edges.

This is a random URL as an example:

```
directory.google.com/example/test.php?key=value&one=1
```

Derived URLs will be as follows:

```
directory.google.com/example/test.php
directory.google.com/example/
directory.google.com/
google.com/example/test.php?key=value&one=1
google.com/example/test.php
google.com/example/
google.com/
```

The resulting URLs must be checked in the database.

Local database

One of the ways to get URL categories is to use the local database provided by SafeDNS. This database is getting updates on a daily basis and therefore the database always contains actual information about websites on the Internet. This database is currently supplied in sqlite3.

Updating the database

The database is provided in two sets: a binary file (sqlite3 base) and a patch (SQL constructions to bring the existing sqlite3 database to the actual state).

Here is the source to download the database in a binary file:

```
https://url2cat.safedns.com/pubfilter/grandbase.db
```

Here is the source to download the database in patches:

```
https://url2cat.safedns.com/api/v1/update/<user_version>
```

Where `<user_version>` is the PRAGMA parameter of the current database version. Inside the resulting patch, the first line indicates the new version of this parameter, and if it matches the requested version, no update is required.

You can check the current version of the database (PRAGMA parameter `user_version`) using the command:

```
xxd -l 4 -s 60 grandbase.db
```

The database can be updated by sending a GET request to the specified source with BASIC authorization parameters.

Database description

The database contains two tables:

1. Table result
2. Table cat

Table result

The table result contains hashed URL entries and their categories.

Table scheme:

Name	Data
domain_hash	Hash of a domain
path_hash	Hash of a path
cat_id	Categories list

With a primary key on the fields `domain_hash`, `path_hash`.

The data in the `cat_id` field is stored as a blob array to standardize the enumerated type, where each category is stored as an unsigned short.

Table cat

The table **cat** contains a list of entries with category names and identifiers.

Depending on customer requirements, the SafeDNS Octo database can be supplied with a different number of categories with more detailed categorization or unique category names.

Table scheme:

Name	Data
locale	localization identifier
cat_id	category identifier
name	category name

With a primary composite key `locale`, `cat_id`

Identifiers from the field `cat_id` of the "result" table are the foreign key to this table. The field `locale` contains the localization identifier of the language in which is written the name of the category in the name field.

The `cat_id` field contains the numerical category identifier, the `cat_id` data is not sequential. The name field contains localized category names.

Example

Along with the source code of the library, a small sample project is supplied, designed to test the functionality of the library. The source code of the project is in `\liburl2cat\example`

Also in this directory, you will find:

- grandbase.db - sample of a SafeDNS database;
- domains.csv - a file containing domains from the sample database
- url2cat.ini - a configuration file that specifies credentials for access to downloading and updating the database, updating paths, etc.

url2cat.ini file contains the following fields:

```
cache_size=0
db_name=grandbase.db
db_update_host=url2cat.safedns.com
db_update_path=/api/v1/update
db_download_path=/pubfilter/grandbase.db
db_update_user=username
db_update_password=password
db_user_version=00000000
server_host=x.api.safedns.com
server_path=/domain
server_user=username
server_password=password
recat_host=www.safedns.ru
recat_path=/api/json/v2
recat_user=username
recat_password=password
```

After configuration and assembly, the `client` file will appear in the `/liburl2cat/build/bin` directory. Move the following files from `\liburl2cat\example` to this directory:

- grandbase.db
- url2cat.ini

The client uses the domains.csv file as the source of domains/URLs for verification. You can create this file from the command line:

```
echo "facebook.com / /" >> domains.csv
```

To start the client, use `./clent` command.

Error Information

When using the url2cat library, errors may occur as a result, for example, lack of Internet connection, etc. Information about these errors can be found in the file

[/liburl2cat/include/url2cat.h](#)

```
#SUCCESS = 0x0000,
/******/
#CACHE_URL_NOT_FOUND = 0x0001,
#DB_URL_NOT_FOUND = 0x0002,
#SERVER_URL_NOT_FOUND = 0x0003,
#URL_END_CHECK = 0x0004,
#DB_IS_BEING_UPDATED = 0x0005,
#WARNING = 0x1000,
/******/
#INIT_REPEATED = 0x1001,
#INIT_DB_NAME_EMPTY = 0x1002,
#INIT_DB_UPDATE_HOST_EMPTY = 0x1003,
#INIT_DB_UPDATE_PATH_EMPTY = 0x1004,
#INIT_DB_DOWNLOAD_PATH_EMPTY = 0x1005,
#INIT_DB_UPDATE_USER_EMPTY = 0x1006,
#INIT_DB_UPDATE_PASSWORD_EMPTY = 0x1007,
#INIT_SERVER_HOST_EMPTY = 0x1008,
#INIT_SERVER_PATH_EMPTY = 0x1009,
#INIT_SERVER_USER_EMPTY = 0x100A,
#INIT_SERVER_PASSWORD_EMPTY = 0x100B,
#INIT_RECAT_HOST_EMPTY = 0x100D,
#INIT_RECAT_PATH_EMPTY = 0x100E,
#INIT_RECAT_USER_EMPTY = 0x100F,
#INIT_RECAT_PASSWORD_EMPTY = 0x1010,
#LOCALE_MALLOC_NULL = 0x1011,
#COMMON_BASE64_LEN = 0x1012,
#COMMON_BASE64_MALLOC_NULL = 0x1013,
#CACHE_SIZE_LIMITE = 0x1014,
#CACHE_BUFFER_MALLOC_NULL = 0x1015,
#CACHE_URL_NO_PATH = 0x1016,
#RECAT_HEADER_MALLOC = 0x1017,
#RECAT_CONTENT_MALLOC = 0x1018,
```



```

[RECAT_REQUEST_MALLOC = 0x1019,
[RECAT_PARSER_ANSWER = 0x101A,
[RECAT_REQUEST_WRITE = 0x101B,
[RECAT_READ_ANSWER = 0x101C,
[URL_INCORRECT = 0x101D,
[CATEGORY_ARRAY_NULL = 0x101E,
[READ_INI_NOT_OPEN_FILE = 0x101F,
[READ_INI_CALLOC_OPTIONS = 0x1020,
[READ_INI_REALLOC_OPTIONS = 0x1021,
[READ_INI_NOT_OPTION = 0x1022,
[READ_INI_NOT_STREAM = 0x1023,
[READ_INI_MALLOC_BUFF = 0x1024,
[READ_INI_OPTIONS = 0x1025,
[/*****/
[CANON_BUFFER_MALLOC_NULL = 0x2001,
[CANON_BUFFER_MALLOC_REPEAT = 0x2002,
[CANON_BUFFER_REALLOC_NULL = 0x2003,
[CANON_CHAR_HEX_UP = 0x2005,
[CANON_CHAR_HEX_LOW = 0x2006,
[CANON_PATH_SLASH_DOT_DOT = 0x2007,
[/*****/
[DB_OPEN = 0x3001,
[DB_JOURNAL_MODE = 0x3002,
[DB_TEMP_STORE = 0x3003,
[DB_SYNCHRONOUS = 0x3004,
[DB_LOCKING_MODE = 0x3005,
[DB_QUERY_CATEGORY_NAME_PREPARE = 0x3006,
[DB_QUERY_CATEGORY_NAME_STEP = 0x3007,
[DB_QUERY_CATEGORY_NAME_NOT_ROW = 0x3008,
[DB_QUERY_CATEGORY_NAME_TYPE = 0x3009,
[DB_QUERY_CATEGORY_NAME = 0x300A,
[DB_QUERY_USER_VERSION_GET_PREPARE = 0x300B,
[DB_QUERY_USER_VERSION_GET_STEP = 0x300C,
[DB_USER_VERSION_GET = 0x300D,
[DB_QUERY_USER_VERSION_SET_PREPARE = 0x300E,
[DB_QUERY_USER_VERSION_SET_STEP = 0x300F,
[DB_USER_VERSION_SET = 0x3010,
[DB_REQUEST_UPDATE_HEAD_MALLOC_NULL = 0x3011,
[DB_REQUEST_UPDATE_FULL_MALLOC_NULL = 0x3012,
[DB_REQUEST_DOWNLOAD_HEAD_MALLOC_NULL = 0x3013,
[DB_REQUEST_DOWNLOAD_FULL_MALLOC_NULL = 0x3014,
```

```
DB_PARSE_ANSWER_NOT_OK = 0x3015,
DB_PARSE_ANSWER_NOT_CONTENT_RANGE = 0x3016,
DB_PARSE_ANSWER_NOT_END_HEAD = 0x3017,
DB_PARSE_ANSWER_NOT_LENGTH = 0x3018,
DB_UPDATE_NOT_MAGIC_STR_HEAD = 0x3019,
DB_UPDATE_NOT_USER_VERSION_HEAD = 0x301A,
DB_DOWNLOAD_NOT_MAGIC_STR_HEAD = 0x301B,
DB_FILE_RENAME = 0x301C,
DB_DOWNLOAD_FILE_NOT_CREAT = 0x301D,
DB_DOWNLOAD_FILE_NOT_FS_STAT = 0x301E,
DB_DOWNLOAD_FILE_NOT_SPACE_DEVICE = 0x301F,
DB_DOWNLOAD_FILE_NOT_FULL = 0x3020,
DB_DOWNLOAD_FILE_NOT_WRITE = 0x3021,
DB_DOWNLOAD_FILE_NOT_MMAP = 0x3022,
DB_DOWNLOAD_FILE_READ = 0x3023,
DB_QUERY_APPLY_PATCH_PREPARE = 0x3024,
DB_QUERY_APPLY_PATCH_STEP = 0x3025,
DB_QUERY_APPLY_PATCH_NOT_DONE = 0x3026,
DB_APPLY_PATCH_NOT_MAGIC_STR = 0x3027,
DB_APPLY_PATCH_NOT_USER_VERSION = 0x3028,
DB_APPLY_PATCH_MISTMACH_USER_VERSION = 0x3029,
DB_APPLY_PATCH_NOT_END_ROW = 0x302A,
DB_UPDATE_HEADER_WRITE = 0x302B,
DB_UPDATE_HEADER_READ = 0x302C,
DB_UPDATE_FULL_WRITE = 0x302D,
DB_UPDATE_FULL_READ = 0x302E,
DB_DOWNLOAD_HEADER_WRITE = 0x302F,
DB_DOWNLOAD_HEADER_READ = 0x3030,
DB_DOWNLOAD_FULL_WRITE = 0x3031,
DB_DOWNLOAD_FULL_READ = 0x3032,
DB_QUERY_CATEGORY_MALLOC_NULL = 0x3033,
DB_QUERY_CATEGORY_PREPARE = 0x3034,
DB_QUERY_CATEGORY_STEP = 0x3035,
DB_QUERY_CATEGORU_NOT_ROW = 0x3036,
DB_URL_NO_PATH = 0x3037,
/*****/
SERVER_REQUEST_GET_PATH_MALLOC_NULL = 0x4001,
SERVER_REQUEST_GET_PATH_REALLOC_NULL = 0x4002,
SERVER_REQUEST_HOST_MALLOC_NULL = 0x4003,
SERVER_REQUEST_BUFFER_MALLOC_NULL = 0x4004,
SERVER_ANSWER_BUFFER_MALLOC_NULL = 0x4005,
```

```
SERVER_ANSWER_NOT_OK = 0x4006,  
SERVER_ANSWER_NOT_CATEGORY_TYPE = 0x4007,  
SERVER_ANSWER_NOT_CATEGORY_TYPE_NAME = 0x4008,  
SERVER_ANSWER_NOT_CATEGORY_TYPE_BRACKET = 0x4009,  
SERVER_ANSWER_NOT_CATEGORY_TYPE_NAME_BRACKET = 0x400A,  
SERVER_ANSWER_NOT_CATEGORY_TYPE_INT = 0x400B,  
SERVER_CATEGORY_NOT_CONNECT = 0x400C,  
SERVER_CATEGORY_HOST_WRITE = 0x400D,  
SERVER_CATEGORY_HOST_READ = 0x400E,  
SERVER_INIT_REPEAT = 0x400F,  
/*****/  
NET_HTTPS_NOT_CONNECT = 0x5001,  
NET_SCHEME_NOT_SUPPORT = 0x5002,  
NET_HOST_PORT_MALLOC_NULL = 0x5003,  
NET_CONNECT_FAILURE = 0x5004,  
NET_WRITE_FAILURE = 0x5005,  
NET_READ_FAILURE = 0x5006,  
NET_CHECK_CONNECT_FAILURE = 0x5007,  
NET_SSL_CONF_CTX_finish = 0x5008,  
NET_SSL_BIO_get_ssl = 0x5009,  
NET_SSL_BIO_do_connect = 0x500A,  
NET_SSL_BIO_do_handshake = 0x500B,
```