SiteFilter™ URL Filtering API

Document Revisions

Revision	Date	Initials	Description
2.1	3/15/2008	JD	Updates to include database updates
2.1.1	4/30/2008	JD	Added url_engine_version, removing custom entries, detail about adding categories and url_debug
2.1.2	5/19/2008	JD	Added url_categories_info. Extended url_init, documented DIA and how to implement direct DIA lookups.
2.1.2.1	5/29/2008	JD	Added category id to category_info struct
2.1.2.2	6/16/2008	JD	Added url_deinit. Updated url_init
2.1.2.3	7/9/2008	JD	Correctly documented port separator encodings
2.1.2.4	7/17/2008	JD	Modified returned error codes in DIA cloud lookup to work around DNS that manipulate return codes.
2.1.2.5	9/5/2008	JD	Formatting change. Added NOTE to section 5.1 regarding checking cert
2.1.3	10/23/2008	JAB	Added reference code section
2.1.4	1/13/2009	JD	Updated perl example to strip all trailing slashes, not just one.
2.1.5	2/11/2009	JD	Show example of label separator.
2.1.6	3/1/2009	JD	Updated category names to be more descriptive
2.1.7	4/6/2009	JD	Added encoding methods
2.1.8	5/12/2009	JAB	
2.1.9	6/8/2009	JAB	Added url_read_custom_category_file API docs and updated url_reload docs.
2.4	7/22/2009	JAB	Added category mapping APIs.

Revision	Date	Initials	Description
3.0	12/1/2009	JAB	Format changes. Updates to url_remove_custom_url.

1. Introduction

eSoft's SiteFilter Web Filtering offering provides software and hardware vendors, as well as service providers, with the most flexible, cost-effective web filtering OEM solution available. The SiteFilter database provides the broadest coverage for the websites your customers visit, the fastest classification rates for new websites, and most comprehensive protection for malicious and dangerous websites.

2. SiteFilter Database

The SiteFilter Database consists of tens of millions of URLs classified into 53 unique categories. eSoft's automatic classification systems and manual classification personnel update the database continuously. The database is delivered through two mechanisms. The first is the optional database stored locally (urldb) and is designed for fast database lookups. Urldb is updated daily and is available through both full and incremental updates. The second mechanism is through eSoft's Distributed Intelligence Architecture™ (DIA) system. This optional hosted service allows classifications of new/unseen URLs to be reported and classified on—the-fly as well as protection against new malicious sites between full database updates. Both mechanisms are covered in this document.

3. SiteFilter Categories

The SiteFilter Database consists of 53 different categories with the ability to add up to 63 custom categories. The built-in categories are defined below:

ID	Category	Description
16	Alcohol	Web pages that promote, advocate or sell alcohol including beer, wine and hard liquor. Includes web pages that give directions on how to make alcohol and mix drinks such as cocktail recipes and home brewing directions. Keywords: liquor, wine, beer, brewery Examples: budweiser.com, coors.com, winespectator.com
12	Anonymizer	Proxies and anonymizers for surfing websites. Tools used to anonymize web use or to circumvent site filters. Keywords: proxy, circumvention, anonymizer Examples: anonymizer.com, torproject.org, anonymsurfen.com

ID	Category	Description
35	Art	Theater, museums, exhibits, photography, artists or photographer, showcases, and digital graphic resources. Keywords: galleries, fine art, cultural institutions, sculpture, opera, symphony, dance Examples: smithsonian.org, galleryworldwide.com, art.com, istockphoto.com, seattlesymphony.org
50	Business/Services	Any business, offers or service. Keywords: industry group, association, marketing, about us, architecture Examples: kinkos.com, proctorgamble.com, architectfinder.aia.org, bbb.org
22	Cars/Transportation	Vehicles including selling, promoting, or discussion. Includes sites that contain information on insuring, repairing, or modifying a car. Excludes sites on trip planning or travel regardless of vehicle. Keywords: motor vehicle, automobile, car, truck, airplane, train, ship, bus, auto insurance, car parts Examples: autotrader.com, harleydavidson.com, amtrak.com, automotive.com, boeing.com
43	Chat/IM	If a site primarily exists for users to chat with each other in real time, then the site is Chat/IM. However, a site that is primarily a Finance site, for example, is not also a Chat site just because it has that extra feature on its site. Also includes sites used to send text messages to phones. Keywords: IRC, SMS, AIM, messaging, live chat Examples: meebo.com, aim.com, ircatwork.com, ebuddy.com
44	Community Sites	Sites dedicated to forums, newsgroups, or bulletin boards. Many sites on a particular topic, such as cars, may have forums, but since there is a more specific category (Cars/ Transportation) this does not apply. This is specifically for communities and forums that don't already have a more specific category. Broad topic mail list archives should also be included in this group. Keywords: forum, message board, list server, community, newsgroup, usenet Examples: well.com, groups.google.com, giganews.com
1	Compromised and Links to Malware	Sites that have been compromised by someone other than the site owner. Includes sites that may be vulnerable to a particular high-risk attack. This includes sites that have been infected or link to malicious sites. These sites are generally classified only by automation using virus checks and other techniques to identify a compromised site.

ID	Category	Description
13	Computers and Technology	Sites with information about computers, software, hardware, peripheral and computer services. Includes sites dedicated to reviewing or marketing technology products. Keywords: web hosting, domains, information security, telecommunications, gadget, high-tech, computer, hardware, software Examples: microsoft.com, esoft.com, engadget.com, dell.com
2	Criminal Skills/Hacking	Activities that violate human rights including murder, sabotage, bomb building, etc. Information about illegal manipulation of electronic devices, encryption, misuse, and fraud. Warez and other illegal software or copy-protected content. Keywords: Terrorism, torture, cracks, crackz, pirated software, warez, copyright violation, evading law enforcement, hacking, lockpicking, burglary, gang, cracking Examples: sectools.org, warez.com, phreak.org, astalavista.com
23	Dating and Relationships	Online dating, matchmaking, relationship advice, personal ads and web pages related to marriage Keywords: personals, singles, matchmaking, romance, dating, mate Examples: match.com, brides.com, marsvenus.com, dearcupid.org
14	Download Sites	Shareware, freeware, and other legal software downloads (see Criminal Skills for pirated software). Includes P2P sites and software, screen savers, web templates, ringtones, and any site with a repository of downloadable items. Keywords: P2P, freeware, shareware, download, free trial, ringtones, wallpaper, personal storage, backup, file sharing Examples: download.com, freewarehome.com, popularscreensavers.com, ringophone.com, kazaa.com
34	Education	Educational institutions, schools, preschools and structured childcare such as Montessori, CCLC, etc). Also educational materials and reference materials including dictionaries, and encyclopedias. Not included: nanny services, baby sitter sites, au pairs. Keywords: history, literature, novels, poetry, research, curriculum, journals, public statistical data, atlas, science, elementary, secondary, high school, university, college, trade school, online courses, map, almanac Examples: harvard.edu, mcgraw-hill.com, dictionary.com, worldatlas.com, redwood.org

ID	Category	Description
36	Entertainment and Videos	Web sites for TV, movies, celebrity sites, entertainment news and gossip sites. Also includes sites devoted to video content, video downloads, or in-browser watching of clips, shows, or movies. Keywords: cinema, fall line-up, movie, videos, celebs, Hollywood, TV, show, clips, ratings, MPAA Examples: oscars.org, eonline.org, youtube.com, nbc.com
40	Finance	Includes businesses dedicated to finance and sites dedicated to discussing finance. Finance includes banking, investing, general insurance (not health, auto, medical, or vision insurance), loans, retirement planning, tax planning, etc. Keywords: interest rates, stocks, bonds, futures, mortgage, taxes, markets, indexes, dow, nasdaq, credit union, credit card, banking, life insurance, liability, online trading Examples: cnbc.com, money.cnn.com, bankrate.com, paypal.com, wellsfargo.com, fidelity.com
20	Gambling	Gambling, lottery, casinos, online poker with real stakes, and betting agencies involving chance. Keywords: odds, sports lines, wagers, betting, pools, lotto, stakes, casino, sportsbook, bookie Examples: gamblingonlinemagazine.com, bodog.com, gambling.com, fulltiltpoker.com
21	Games	Computer games, game producers, cheat sites, game walk-throughs, online games, puzzles, crosswords, etc. Keywords: video games, role-playing, online games, puzzles, cards, board games Examples: nintendo.com, games.yahoo.com, bestcrosswords.com
29	Government	Government organizations, departments, or agencies. Includes police, fire, and hospitals. Government sites may be multiply-categorized for example with Travel. Keywords: emergency services, bureau, town council, local government, city of, house of representatives, senate Examples: whitehouse.gov, ci.boulder.co.us, denvergov.org, iaff858.org
3	Hate Speech	Extreme right/left wing groups, sexist remarks, racist remarks or racial slurs, religious hate, or the promotion of racist or bigoted views. Keywords: bigot, race, nationality, gender, age, disability, slavery, racial slurs, white supremacy, aryan power Examples: whitepower.com, micetrap.net, aryan-nations.org, racist-jokes.com

ID	Category	Description
17	Health	Personal health and medical services including sites with information on equipment, procedures, mental health, finding doctors, recovery (drug and alcohol), etc. Does not include prescription drug information. Also includes information on doctors and medical insurers. Keywords: self-help, nutrition, diet, counseling, psychology, nurse, addiction, therapy, yoga, chiropractic, medical insurance, doctor, hospital, dentist, optometry, cancer, disease Examples: webmd.com, health.com, jeffersonhospital.org, ahd.com
24	Home/Leisure	Sites with information about home improvement, decorating, family, pets, gardening, cooking, event planning (non-wedding) and hobbies. Also includes babysitting services, au pairs, adoption, etc. Keywords: home improvement, recipe, garden, decorate, hobby, crafts, family, adoption, family pets Examples: marthastewart.com, rc-airplane-world.com, babysitters.com, doityourself.com
37	Humor	Comedians, comic strips, jokes, and other humorous content or diversions. Keywords: comics, comedian, comic strip, jokes, stand-up, funny Examples: comedycentral.com, dilbert.com
4	Illegal Drugs	Use or information on commonly illegal drugs, misuse of prescription drugs, and other compounds. Includes sites giving non-clinical descriptions or stories about being high. Also includes sites on legalizing marijuana and sites that display pictures of marijuana plants if shown in a way that could be considered an endorsement of the drug. Keywords: addiction, LSD, heroine, cocaine, dope, amphetamines, stimulants, marijuana, mary-jane, pot, drug abuse, meth Examples: totse.com, marijuana.org, www.methspace.com
51	Job Search	Job posting sites, career planning, human resources, interview tips, resume help, classified ads for jobs, and other sites about searching for a new job. Keywords: Headhunter, recruiter, curriculum vitae (CV), resume, employment, career, hr, interview, job Examples: monster.com, careerbuilder.com, jobweb.com, headhuntersdirectory.com

ID	Category	Description
8	Mature	This category is for sites with adult content that do not already fit under another adult category. This includes sites that discuss alternative lifestyles, sex/sexuality, pregnancy, and abortion. Also includes sites that use profanity, are offensive, disgusting, or tasteless. Keywords: abortion, cloning, homosexual, gay, lesbian, sexuality, sex education, pregnancy, birth control, profanity Examples: realsexedfacts.com, teensource.org, plannedparenthood.org, gaypride.com
30	Military	Sites sponsored or devoted to the armed forces and government controlled agencies. Keywords: Navy, Air Force, Marines, Army, Seals, Commando, Special Forces Examples: janes.com, army.mil, navy.mil, defenselink.mil
54	Miscellaneous	Websites that do not have any content that can be found, contain blank or missing pages or that just do not fall into any other category. Also sites like Akamai that host data on behalf of so many companies that they probably could have every category applied.
38	Music and Audio	Internet radio, streaming media, musicians, bands, and MP3 download sites. Includes music videos, but does not include sites for downloading ring tones, etc., use Download Sites for these items. Keywords: music, band, group, choir, mp3, radio, rock, easy listening, classical Examples: kygo.com, itunes.com, rollingstone.com, billboard.com
39	News and Magazines	General news information such as found in newspapers and magazines. Includes weather, but does not include thematic news that fits under another category such as Finance or Entertainment. Keywords: media, magazine, news tabloid, publication, weather Examples: cnn.com, nytimes.com, weather.com, news.google.com
31	Non-profits	Clubs, communities, unions, and non-profit organizations. Includes fraternities and sororities. Does not include sites that have a more specific category. Keywords: philanthropy, boy scouts, 4H, rotary club, volunteer, not for profit, charity Examples: 4husa.org, uswa.org, charitywatch.org, iicenter.org

ID	Category	Description
7	Nudity	Photographs or drawings showing a woman's bare breast or butt. In rare cases where the intent is clearly not offensive or sexual photographs that show genitalia may be categorized as nudity. Sites with photographs containing nearly nude models wearing only underwear or lingerie should also be considered Nudity. Does not include sites where the focus is specifically on the butt, breast, or crotch even if these parts are covered. Such sites should be classified Pornography. Keywords: lingerie, nudist, pin-up, intimate apparel, negligee, nude art, naked, sexy Examples: victoriasecret.com, fineartnude.com
49	Online Ads	Sites responsible for hosting online advertisements including advertising graphics, banners, and pop-up content. Keywords: advertising, online marketing Examples: doubleclick.net, adbrite.com
25	Personal Webpages	Sites about or hosted by personal individuals. Includes personal blogs, guestbook servers, people talking about their family, personal hobbies, activities, etc. Keywords: blog, homepage Examples: geocities.yahoo.com, healthypet.com, blogger.com, homepages.aol.com
18	Pharmaceuticals	Prescribed medications and information about legal drugs, drug studies, medical use, or pharmacies (online or otherwise). Keywords: medicine, prescription, pharmaceuticals, drugstore Examples: walgreens.com, claritin.com, drugs.com
5	Phishing and Fraud	Manipulated websites and emails for fraudulent purposes, also known as phishing. These sites are often trying to impersonate legitimate sites in order to steal passwords, social security numbers, and other confidential information.
32	Politics and Law	Sites that promote a political party, interest group, or issue. Information on elections and legislation, legal firms, legal information and advice. Also sites with codes and laws. Keywords: advocacy, lobbyist, lawyer, activist, environmental protection, law, red state Examples: aclu.org, democrats.org, nolopress.com

ID	Category	Description
9	Pornography/Sex	Explicit content including text or images depicting a sexual act, sexual arousal, or nude images intended to be sexual in nature. May also include explicit photos without nudity such as close-up pictures of body parts such as women's cleavage (even though the woman may be wearing a bra). Keywords: erotic, strip, sex, porn, xxx, adult film, exhibitionism, nymph, milf, up-skirt, down-shirt, voyeur, virgin, fetish, horny, orgy Examples: playboy.com, freehorny-xxx.com, upskirt.com, freeporn247.org
47	Portal Sites	General sites with customizable personal home pages, parked domains, and sites that contain only links and no content of their own. Keywords: parked domain, domain for sale, portal, customizable home page Examples: my.yahoo.com, my.aol.com
52	Real Estate	Information about renting, purchasing, selling, or financing real estate including homes, apartments, condominiums, office space, etc. Includes sites listing property for sale and real estate agents/brokers. Keywords: MLS listing, real estate, townhouse, mortgage, interior design, property, architecture, land, appraisal Examples: remax.com, cohomefinder.com, zillow.com
33	Religion	Religious sites and information including churches, bibles, sects, and occultism. Keywords: church, mosque, synagogue, sect, cult, Buddhism, bahai, Christian, hindu, islam, mormon, shinto, atheism, horoscope Examples: catholic.net, bible.com, fpcboulder.org, allaboutreligion.org
26	Restaurants	Food, dining, catering services, food critics, restaurant reservation and restaurant review sites. Keywords: bar, fast food, pub, cafe, restaurant, reservations Examples: chilis.com, restaurants.com, opentable.com
48	Search Engines	Sites supporting the searching of web, newsgroups, pictures, directories, and other online content. Also white/yellow pages. Keywords: search Examples: yahoo.com, google.com, ask.com, whitepages.com

ID	Category	Description
42	Shopping	Online shops, catalogs, and online ordering. Auction sites and classified ads that are selling something other than Jobs, Autos, or Real Estate. Excludes shopping for products and services exclusively covered by another category such as Health. Keywords: clothing, cosmetics, fashion, checkout, perfume, jewelry, collectors, antiques, department store, retail, merchandise Examples: amazon.com, ebay.com, pricegrabber.com, overstock.com
45	Social Networking	Social Networking sites are web sites in which users have their own space to talk about themselves. It is also characterized by the ability to link to other people and control what information is public versus what information can be seen by members who are linked. Keywords: friends, network, profile Examples: myspace.com, friendster.com, linkedin.com, bebo.com
53	Spam URLs	Web site links found inside spammed e-mails are automatically added to this list.
27	Sports and Recreation	Sports teams, fan clubs, and sports news. Also includes recreational activities including zoos, public recreation centers, pools, amusement parks, skiing, skydiving, etc. Keywords: sport, league, Olympics, theme parks, camping, hiking, rock climbing, fantasy sports, SCUBA, scores, soccer, basketball, football, boxing, martial arts Examples: nba.com, denverbroncos.com, vailresorts.com, sixflags.com
6	Spyware and Malicious Sites	Sites or software that installs on a user's computer with the intent to collect information or make system changes without the user's consent. Includes sites that try to trick users into installing software or try to exploit vulnerabilities in web browsers to install software automatically. Keywords: spyware, Trojan, backdoor, keystroke logger, malware, virus, exploit Examples: (none)
19	Tobacco	Sites that promote or sell tobacco products such as cigarettes, cigars, and chew. Keywords: cigar, cigarette, pipe, chew Examples: cigar.com, marlboro.com, phillipmorris.com
15	Translator	Translate sites from one language to another. Keywords: translate, language Examples: translate.google.com, babelfish.altavista.com

ID	Category	Description
28	Travel	Provides travel and tourism information, including all sites devoted to planning trips, bus/train/airplane timetables and routes, or online booking or travel services such as car rentals, hotel booking, etc. Includes regional or city information sites. Keywords: cruise, destinations, travel, beach, monuments, resort, guide, hotel, flight, trip, vacation Examples: travelocity.com, orbitz.com, aaa.com, frontierairlines.com
10	Violence	Instructions on how to commit violence excluding Criminal Skills like bomb making. Includes militancy, torture, crime-scene photos, and descriptions or pictures of a violent or gory nature. Also includes sites that promote violence. Keywords: bodily harm, gore, death, injury, mutilation, crime-scene, militancy, anti-government, rebel, sabotage, militia Examples: crimescene.com, deathnet.com, michiganmilitia.com
11	Weapons	Guns and weapons when not used in a violent manner such as descriptions, sport hunting, gun clubs, or paintball. Also includes other weapons like crossbows, knives, etc. Keywords: knives, guns, weapons, wargames, paintball, gun clubs, crossbow, swords Examples: historicalarms.net, hunting.net, nra.org, glock.com, weaponmasters.com
46	Web-based Email	Enables users to send and/or receive email through a web accessible email account. Also includes sites that send postcards and greetings to other users. Keywords: mail, e-mail, webmail, online postcard Examples: mail.yahoo.com, hotmail.com, mail.google.com, webmail.us

SiteFilter SDK Implementation Details

The following sections describe the APIs needed to implement SiteFilter URL Classification Engine on Linux, BSD and Windows (additional OS support coming soon).

Dependencies

The SiteFilter SDK depends on certain libraries and command line utilities. These must be installed on the system running the SiteFilter SDK.

3.1.1. cp

The SDK must make intermediate copies of urldb when applying incremental updates. On Unix like systems the cp command is used for this. Windows uses the CopyFile function.

3.1.2. gzip

The full versions of the urldb database are gzip compressed. To uncompress these files the SDK uses the gzip program.

3.1.3. xdelta

For applying the urldb incremental updates, the SDK uses the xdelta program. Note that this is xdelta version 1, not version 3.

3.1.4. curl

The SDK uses curl to download updates. The SDK may use libcurl linked to libsitefilter.so or it may use the independent curl program. This depends on which type of SDK you are using.

3.2. Using the SiteFilter URL Classification Engine

3.2.1. Setting SiteFilter Configuration Options

```
int url_option(const char *option, const char *value)
```

`option` - Name of a SiteFilter option.
`value` - String with the value for the option.

```
int url_option_file(const char *fname)
```

`fname` - Passing NULL will make it read .sitefilterrc in $HOME or in the current directory.

Option Name	Option Description
path	The directory path where the urldb files are stored. If this is set before calling url_init, the path argument to url_init may be NULL.
dia-host	The host name used in DIA DNS queries. If this is set before calling url_init, the dia_host argument may be NULL.
dia-serial	The DIA serial number. If this is set before calling url_init, the dia_serial argument may be NULL.
dia-timeout	The number of seconds for url_lookup to wait for each DIA query response. The default value is ‘0’ which means url_lookup does not wait at all, but returns Uncategorized or categories inherited from the parent.

Option Name	Option Description
default-mapping	The name of the category mapping to use when returning category results.
update-url	The URL to be used when doing url_update. If this is set before calling url_update, the url argument may be NULL.
proxy-url	The URL of the proxy to be used when doing url_update.
proxy-user	The name of the proxy user when authentication is needed.
proxy-pass	The proxy password when authentication is needed.
proxy-method	The proxy authentication method.
debug-level	The debug level. May be set from 0 to 9.
debug-output	Where the debug output is sent. May be stderr or syslog.

3.2.2. Initializing the SiteFilter Database

```
int url_init(const char * path_to_db_files, int verify_db, const char *
dia_dns_host, char * serial_number)
```

`path_to_db_files` - Absolute path containing the SiteFilter database and configuration files without trailing slash

`verify_db` - Specifies whether you want the SiteFilter URL Classification Engine to verify the main database. 1= yes (recommended), 0 = no (decreases initialization time) Verify is always performed on customdb and diadb.

`dia_dns_host` = Specifies the hostname used for the DIA real-time lookup interface. If unset or null, DIA live lookup will not be used.

NOTE: You must store DIA key in file path_to_db_files/dia.key. dia.key may also be placed on download_url to be automatically retrieved during url_update (specified in section 4.2.8).

`serial_number` = specifies the unit serial number used for licensing. If unset or null, DIA live lookup will use standard lookup and licensing. This is used in per-device license enforcement of DIA lookups.

url_init returns 0 on success, non-zero on error

NOTE: The database will be checked for expiration during `url_init`. The expiration time is 60 days. If `url_update` has not been used to download a new database in that time, the urldb cannot be used.

3.2.3. Shut Down the SiteFilter Database

```
int url_deinit(void)
```

`url_deinit` returns 0 on success, non-zero on error

3.2.4. Reload the SiteFilter Engine and databases

```
int url_reload(void)
```

Returns 0 on success, non-zero on error

Reinitialize the SiteFilter API library, move any *.new files into place and read in *.db files. Use after downloading updates or after manually placing new update files in place with *.new names.

The reload is thread-safe and affects all threads in the process. It will also be detected and `url_reload` will be done by other processes accessing the same database directory.

3.2.5. Querying the SiteFilter Database

```
int url_lookup(const char * url, int result[5])
```

```
int url_lookup_match(const char * url, int result[5], char * match,
size_t match_len)
```

```
int url_lookup_cache(const char * url, int result[5], char * match,
size_t match_len)
```

```
int url_lookup_map(const char * url, int result[5], int map_id)
```

```
int url_lookup_match_map(const char * url, int result[5], char * match,
size_t match_len, int map_id)
```

```
int url_lookup_cache_map(const char * url, int result[5], char * match,
size_t match_len, int map_id)
```

The `url_lookup` function takes a pointer to the URL string and returns up to 5 result categories using the int[5] array pointer.

The `url_lookup_match` function also takes the pointer and length of a character buffer and returns the best match found in the URL database. For example, a lookup of "www.esoft.com/images/header/quote_header4.jpg" would return "esoft.com" in the `match` buffer. A leading dot on this string indicates that the URL database has more entries available for that URL path.

The `url_lookup_cache` function also takes the pointer and length of a character buffer and returns the best URL found for caching the result. For example, a lookup of

"www.esoft.com/images/header/quote_header4.jpg" would return "esoft.com" in the `match` buffer. There is no leading dot because the URL database does not contain any entries for child URLs of esoft.com. However, take "intel.com" as an example. Our database also includes entries for "intel.com/reseller" and "intel.com/jobs". A lookup of "www.intel.com/reseller/images/hdr_mapreseller.gif" would result in a cache URL of "intel.com/reseller". A lookup of "www.intel.com/notexist/images/hdr_mapreseller.gif" would result in "intel.com/notexist" even though the "/notexist" path is not in our URL database. That extra path is added in order to avoid using ".intel.com" as the cache. The leading dot shows that "intel.com" cannot be relied on to categorize "intel.com/reseller" or any other child item. The cache items "intel.com/reseller" and "intel.com/notexist" have no leading dot and are reliable because there are no URL database child items of those URLs.

The `url_lookup_map` and `url_lookup_match_map` and `url_lookup_cache_map` functions also take a map_id argument which changes the mapping of the category IDs returned in the `result` argument. Use the `url_get_mapping_id` function to get a `map_id` value from a mapping name.

A positive number returned is the number of categories. A negative return indicates an error.

NOTE: The database will be periodically checked for expiration during lookup operations. The expiration time is 60 days. If `url_update` has not been used to download a new database in that time, the urldb cannot be used.

3.2.6. Decoding errors received from SiteFilter

```
const char *url_errstr(int errorcode)
```

The `url_errstr` function takes an error code as returned from `url_init` or `url_lookup` and returns a character string with a short description of the error.

3.2.7. Retrieving database version from SiteFilter

```
int url_db_version (void)
```

The `url_db_version` function returns the version of the initialized database. The function returns a positive integer representing the version of the database and a "-1" on error or if database is uninitialized.

3.2.8. Retrieving the version of the SiteFilter Engine

```
int url_engine_version (char *version, size_t length)
```

The `url_engine_version` functions returns an integer representing the major and minor version of the engine. The major version is represented above one thousand and the minor version is represented below one thousands. For example, 2005 is major version 2

and minor version 005 or version 2.005.

`version` – string containing the version

`length` – length of version string

3.2.9. Choosing Proxy Authentication

```
enum url_update_proxy_auth_method {
    URL_PROXY_DEFAULT,
    URL_PROXY_BASIC,
    URL_PROXY_DIGEST,
    URL_PROXY_NTLM,
    URL_PROXY_ANY
};

int url_update_proxy_auth(const char *user, const char *pass, int
method);
```

The `url_update_proxy_auth` function changes the proxy authentication used by curl in the `url_update` process. The user name and password will be overridden by any user name and password provided in the proxy URL. NULL values or empty strings may be passed for both user and password.

The default authentication method first tries Basic, then Any if Basic fails. The Any method first tries the proxy without any authentication and examines the results, then uses the most secure of the proxy's supported options. Older versions of the cURL library have bugs in their NTLM authentication. Curl 7.15 is known to work.

3.2.10. Updating the SiteFilter Database

```
struct extra_info {
      const char *key;
      const char *value;
};

int url_update(const char *download_url, const char *proxy_url, int
method, struct extra_info[])
```

`extra_info` - null terminated struct, passes data in GET or POST parameters depending on method used for vendor tracking

optional parameter keys:

- product_name (vendor)
- product_version (vendor)
- os
- user_count
- serial_number

any other vendor deems appropriate.

`download_url` can encode username/password info, protocol, etc. download_url is the

parent directory for the required files. Supported authentication methods: Auth-Basic.

`proxy_url` can encode the username/password, host and path into url format. Leave unset or NULL if not needed.

`method` – GET=0, POST=1

NOTE: Requires `url_init` even if url_init returned error. Requires `url_reload` after download succeeds.

NOTE: The database will be periodically checked for expiration during initialization and lookup operations. The expiration time is 60 days. If `url_update` has not been used to download a new database in that time, the urldb cannot be used.

3.2.11. Monitoring the Update Process

```
struct url_status_info {
      time_t start_time;
      time_t estimated_time;
      double current_bytes;
      double total_bytes;
      time_t op_start_time;
      char *current_op;
      char *current_file;
      double current_file_bytes;
      double total_file_bytes;
      char buffer[128];
};

int url_update_status(struct url_status_info *info);
```

The `url_update_status` function fills in an information structure with data about the currently running update process.

This function can be called once every second or two to update a progress display. For an event driven interface, place a file update monitor on the file named "`update-status.txt`" in the database path.

All of the byte counts refer to bytes processed, not only downloaded. The update process has many steps. Some of the most time consuming steps may be the incremental patching, so those get byte counts of their own.

The byte counts are in floating point to avoid 32-bit integer overflow. An incremental patch update of five or six patches may use byte counts of five gigabytes and more.

All of the strings are stored in `buffer` and pointers to the individual strings are provided for easy access.

`start_time` is the time when the update process started.

`estimated_time` is the guess at when the update will be finished. This guess is not very accurate and can vary widely with changes in download speed and is especially variable

when doing incremental updates.

`current_bytes` is the number of bytes already processed.

`total_bytes` is the number of bytes in the entire process. This is an estimate because the final uncompressed sizes of patched or downloaded databases is unknown.

`op_start_time` is the time when the current operation started.

`current_op` is a C string containing a short description of the current operation such as "downloading", "unzipping", "patching".

`current_file` is a C string with the name of the current file being operated on.

`current_file_bytes` is the number of bytes in the current file.

`total_file_bytes` is the expected number of bytes in the current file when it is complete. This is an estimate.

`buffer` should not be used. It is just the space used for storing string data.

3.2.12. Adding Custom Sites to SiteFilter From Files

There are cases when users will want to add their own sites and classifications or will want to override classification from the main database. Adding custom sites allows users to perform both of these functions.

Lookups

The first step to adding custom sites is to write those site and categories to `/path_to_db_files/your_custom_list.urls` in the following format:

url <TAB> cat_id,…,cat_id<LF> (up to 5 category ids – 3 standard and 2 custom)

Once the file is written, the SiteFilter Engine can be instructed to load the file using the following API:

```
int url_read_custom_files(void)
```

`url_read_custom_files` reads all files with *.urls extension from `path_to_db_files` to incorporate into the custom database

`/path_to_db_files/custom.urls` is used to store entries added with `url_add_custom_url`. Do not use this file name to write your own entries

`/path_to_db_files/customdb` is the binary database generated from the text files and used by the engine for fast queries of custom entries.

3.2.13. Adding Custom Sites to SiteFilter One At A Time

```
int url_add_custom_url(const char * url, const int categories[5])
```

`url` – full url including, protocol, domain, port and path to be added to the custom list database

`categories` – array of up to 5 categories (3 standard and 2 custom) to be associated with url. Use 0 for unused category array items.

`/path_to_db_files/custom.urls` is used to store entries added with `url_add_custom_url`

NOTE: `url_reload` is not required

3.2.14. Adding Custom Site Lists

```
int url_read_custom_category_file(const char *file, const int
categories[5])
```

`file` – The name of a file containing one URL per line. If the file name is not absolute (does not start with / or \ or X:/) then the file should be in the directory provided to `url_init`.

`categories` – array of up to 5 categories to be associated with the URL. Use 0 for unused category array items.

3.2.15. Removing Entries from Custom Site Files

To remove entries from custom sites, remove the entry from all of the *.urls files in `path_to_db_files`, such as `custom.urls`, and call `url_read_custom_files`.

3.2.16. Removing Custom Sites One At A Time

```
int url_remove_custom_url(const char *url)
```

`url` - full URL including protocol, domain, port and path to be removed from the custom URL database.

This will remove one entry from the binary custom site database but not from the *.urls files. This means that if the binary database becomes corrupt or if `read_custom_files` is called, the entry may be added back.

3.2.17. Associating category identifiers with category names

```
int url_categories_name(const int categories[5], const char *delimiter,
size_t delim_length, const char *catnames, size_t *len_catnames)
```

`categories` – array of categories to convert to names

`delimiter` – character(s) used to delimit category names in returned string

`delim_length` – length of delimiter used above

`catnames` – string containing list of category names

`len_catnames` – size of `catnames`. Modified to contain the return string length size

Category names are read from `/path_to_db_files/category.txt` which is in the

following format:

Cat_id<TAB>category_name<TAB>category description<LF>

`url_categories_name` returns 0 on success, non-zero on error

3.2.18. Retrieving all categories

```
struct category_info {
      int   id;
      const char *name;
      const char *description;
};
```

```
char* url_categories_info(struct category_info **info, size_t
*info_count)
```

`info` – Pointer to pointer to array of `category_info` structures, allocated by the function. Pass it a pointer to a pointer in the calling function.

`info_count` – A pointer to the number of `category_info` records.

Call this function and it will allocate the necessary memory and fill it with `category_info` records. When you are done with the information, call "free" on the returned pointer.

3.2.19. Adding custom categories

Adding custom categories requires modifying `/path_to_db_files/categories.txt` using the following format:

Cat_id<TAB>category_name<TAB>category description<LF>

Custom cat_id must be between the numbers of 101 and 163. Categories 0-63 are reserved for the categories listed in section 2 and future use.

NOTE: `categories.txt` must be in either ASCII , UTF8 or ISO 8859 format for `url_categories_name` to return a category name.

3.2.20. Reporting misclassified sites through DIA

```
int url_dia_report(const char *url, int suggested_cats[3])
```

`url` – domain and path to be submitted

`suggested_cats` – array of category id to be suggested (up to 3 standard categories)

Occasionally end-users may disagree with a classification of a URL. In those cases, eSoft has supplied the means to allow users to submit those suggestions directly to eSoft through eSoft's DIA system. The reports are sent directly to manual classifiers to be reviewed.

Submissions do not guarantee change.

This requires the `dia_dns_host` parameter to be set and valid in `url_init` and it requires a current and valid DIA Key.

If the vendor has arranged with eSoft to use their own choice of host name for `dia_dns_host` then the submission data will appear to go to the vendor and not eSoft.

3.2.21. Enabling debug services

```
void url_debug(int level)
```

`url_debug` is used to set the level of debug message sent to STDERR.

0 (default) is no messages console.

9 is all messages sent to console

3.2.22. Changing debug output destination

```
void url_debug_function( void (*f)(const char*, size_t n) )
```

This function accepts the address of another function which takes a character string pointer and a length. That function may do whatever it likes with the debug information that will be sent to it. Debug information will usually be sent one line at a time with newlines included, but that is not guaranteed.

The default function writes debugging output to STDERR.

3.2.23. Use an alternate category mapping

```
int url_get_mapping_id(const char *map_name)
```

This function gets a `map_id` value for use with `url_lookup_map` and `url_lookup_match_map`.

It returns a negative error code.

3.2.24. Select default category mapping

```
int url_default_mapping(const char *map_name)
```

This function sets the default category mapping to the chosen map. These maps are found in the database directory with names ending in .map. The `map_name` is the file name, not including .map. For example, the sitefilter-v2.map file provides the sitefilter-v2 mapping.

It returns either a negative error code or the previous default `map_id`. It can be called with a NULL pointer as `map_name` in order to get the default `map_id` without changing it.

The default category map affects all API functions that do not use a `map_id` argument. This includes DIA submissions and all custom URL functions.

3.2.25. Flush the DIA URL cache

```
int url_dia_cache_flush(void)
```

This removes all of the URLs received through DIA and cached by the SiteFilter library. This also happens whenever `url_reload` is called after a new urldb file is downloaded by `url_update`.

4. Distributed Intelligence Architecture™ (DIA) for Direct Cloud Queries

The system used for delivering URL categorizations in real time from in the cloud. Also used to collect unclassified and misclassified URLs and for automatically classifying URLs.

The URL lookup component of DIA uses the DNS protocol to fetch the categories for a given URL. This allows the lookups to be made through firewalls, to use built-in caching across the Internet, and to use light-weight UDP packets for communications. The DNS lookups are protected by a rotating key as explained below.

Users of the SDK do not need to implement or understand this protocol as it is incorporated into the SDK and is automatic if enabled.

4.1. DIA DNS Rotating Secret Key Retrieval

OEMs will use this method to fetch the secret key and will provide this key to their devices.

https://subscriptions.esoft.com/api/GetRBLKey.php?OEMKey=<oemid>

OEMKey is a key provided by eSoft to OEM for access to DIA. The Key will rotate weekly and must be updated weekly on the devices. DIA will store the previous 3 weeks of keys to ensure clients that fail to retrieve a key will have time to re-request before service is disabled

`oemid` will be provided to OEM by eSoft, Inc.

`GetRBLKey.php` returns a string containing the rotating secret key or "Error: <reason>"

NOTE: Be sure to enforce certificate validation checks on https calls to subscriptions.esoft.com. Man in the middle attacks could result in theft of your OEM Key.

4.2. Using DIA to Retrieve Categories

URL categorizations are retrieved using a standard DNS TXT query. The query will return a list of category ids and the suggested url for caching purposes if the URL is categorized, a domain not found error (NXDOMAIN) if the site is not yet categorized, or REFUSED if the system is not authorized to use the system. Uncategorized sites are pushed into the auto-classification system and can be categorized and ready for the next customer within minutes.

4.2.1. Building the query

The URL being queried must first be encoded as described below. The general form of DNS query looks like this:

<Category>.<EncodedURL>.<AuthString>.<OEMPrefix-SerialNumber>.<Server>

Ex: 0.www.domain.com_-.index.html.01EF.aa-50534.url.esoft.com

where 01EF is the AuthString, url.esoft.com is the lookup domain (may be branded for different OEMs), aa is the OEM's assigned prefix, 50534 is the unit's serial number, and www.domain.com_-.index.html is the encoded URL.

Detailed TXT Record Format:

<Category> = use 0 if the category of the URL is unknown

To report a misclassified URL, put the suggested category ID here. Use underscores to separate multiple categories (up to 3). Ie
3_7_9.<EncodedURL>.<AuthString>.<Serial>.<Server>

<url > = normalized/encoded URL (See section Encoding URLs)
<AuthString> = this string is used to authenticate the request using a combination of the serial number, a one-way hash of the URL, and the Secret Key downloaded by the OEM from DIA using the interface specified above. The calculation of the AuthString is described in more detail below.

<OEMPrefix-SerialNumber> = The OEM prefix is a preassigned two character string associated with a given OEM partner. This is used to avoid serial number collisions between vendors. The serial number is the local serial number of the appliance. An OEM may choose to use a single serial number across all appliances, deliver an authorization code to appliances that are used here, or anything else. eSoft will compare the serial number to a list of allowed serial numbers supplied by the OEM and return a DNS REFUSED error if the serial number is not authorized.
<server> = branded OEM domain, eg, dia.oem.com, which points to eSoft's DIA servers as the authoritative server for that subdomain.

4.2.2. Encoding URLs

To encode URLs, the following rules must be used. Failure to properly encode URLs will likely lead to inability to submit to DIA:

Remove all trailing slashes from URL

Remove any single quotes

Remove all characters passed as arguments. This involves removing all characters after a question mark ("?") character or a hash ("#") character in a URL.

The path separator slash ("/") character should be replaced with the legal DNS character string underscore, dash, dot ("_-.").

Port numbers separator colon (":") character should be replaced with the legal DNS character string underscore, dash, dash ("_--")

Encode all illegal DNS characters into their URL percent encoded representation. For

example, a tilde (“~”) character needs to be percent encoded to “%7e”. The percent character must be converted into underscore dash format to be submitted to DIA as described in the next point.

Encode characters used for URL encoding. This is done by converting % to underscore, dash (_-).

Take care of case where _- character sequence exists in URL. This is done by converting _- to _-_-.

DNS prohibits two empty labels (two dots next to each other). If the URL ends is a dot, it must be percent encoded.

DNS requires labels (sections between dots) to be less that 64 characters. If you have a part that exceeds that, you must encode with an underscore, dash, zero, dot (_-0). This requires you to check at 60 bytes to insert this encoding.

Length of entire query must not exceed 255 characters. If length of encoded category + url + serialnumber + server + 7 > 255, you must truncate encoded url.

AuthString:

The hex representation of a 16 bit CRC-CCITT hash of the following concatenated strings:

“domain” + “dia_key” + “OEMprefix-serialnumber”

NOTE: domain contains the subdomain and domain portion of the url without any path for the purpose of caching CRC calculations for future requests.

Return Values:

NXDOMAIN if accepted but not categorized

ANSWER (see below)

Answer Format:

When DNS responds with an ANSWER, it returns a TXT record with one of the following formats:

Category found:

categoryId_categoryId<TAB>cache url<LF>

That is an underscore-separated list of category id’s followed optionally by a tab character and a cache URL. This cache URL is returned when a local database can cache the current URL using a given cache URL and can assume that all URLs under that base URL are the same category. In the case where a top-level domain is categorized, but subdomains or subpaths have different categorizations, The base URL is returned with a leading dot (“.”) indicating only that exact URL can be cached and URLs under it should

be queried.

NOTE: In case of error, a negative value is returned for a category ID.

Invalid AuthString:

FAILURE: Invalid authorization

Invalid SerialNumber:

REFUSED: Unlicensed serial number

NOTE: Additional lines returned are reserved for future use. Develop software to only use first line in response and ignore all additional lines.

Example of transmission

Retrieve Secret Key from SoftPak Director to communicate with DIA:

https://subscriptions.esoft.com/api/GetRBLKey.php?OEMKey=<key>

Returns: 523008891

Assume a local OEM prefix of "aa" and a local serial number of "12345".

Submit the following URL to DIA:

http://www.domain.com/Site/Resources/Check_something.asp?arg1=127.0.0.1

First, encode the URL:

www.domain.com_-.Site_-.Resources_-.Check_something.asp

Next, take the CRC-CCITT value of the concatenated string:

0x12AC = crcccitt("www.domain.com523008891aa-12345)

So to query this URL you would use a command like this:

dig 0.www.domain.com_-.Site_-.Resources_-.Check_something.asp.12AC.aa-12345.url.esoft.com

Perl Example

```
#!/usr/bin/perl
# Copyright 2008, eSoft, Inc.
# eSoft Confidential

use Digest::CRC qw(crc32 crc16 crcccitt crc);
use LWP::UserAgent;
use HTTP::Request::Common;
use Net::DNS;

%categories= (
49 => 'Adware',
16 => 'Alcohol',
12 => 'Anonymizer',
```

```
35 => 'Art',
50 => 'Business/Services',
22 => 'Cars/Transportation',
43 => 'Chat/IM',
44 => 'Community Sites',
1 => 'Compromised',
13 => 'Computers and Technology',
2 => 'Criminal Skills/Hacking',
23 => 'Dating',
14 => 'Download Sites',
34 => 'Education',
36 => 'Entertainment and Videos',
40 => 'Finance',
20 => 'Gambling',
21 => 'Games',
29 => 'Government',
3 => 'Hate Speech',
17 => 'Health',
24 => 'Home/Leisure',
37 => 'Humor',
4 => 'Illegal Drugs',
51 => 'Job Search',
8 => 'Mature',
30 => 'Military',
54 => 'Miscellaneous',
38 => 'Music',
39 => 'News',
31 => 'Non-profits',
7 => 'Nudity',
25 => 'Personal Webpages',
18 => 'Pharmacy',
5 => 'Phishing/Fraud',
32 => 'Politics and Law',
9 => 'Pornography/Sex',
47 => 'Portal Sites',
52 => 'Real Estate',
33 => 'Religion',
26 => 'Restaurants',
48 => 'Search Engines',
42 => 'Shopping',
45 => 'Social Networking',
53 => 'Spam',
27 => 'Sports and Recreation',
6 => 'Spyware and Malicious Sites',
19 => 'Tobacco',
15 => 'Translator',
28 => 'Travel',
10 => 'Violence',
11 => 'Weapons',
46 => 'Web-based Email');

sub encodeURL($)
```

```
{
      my ($query) = @_;

      my ($domain, $port, $path) = $query =~
            m!^(?:[^:/]*://)?([^:/\\]*)(:\d+)?([/\\][^?#]*)?!;
      ## Remove any trailing dots
      $domain =~ s/\.$//;
      $query = $domain;
      $query .= $port if $port;
      $query .= $path if $path;

      ## Remove trailing slashes
      $query =~ s/\/+$//;
      ## Take care of the rare, but possible case of _-
      ## being in the string
      $query =~ s/_-/_-_-/g;
      ## Convert / to rfc compliant characters _-.
      $query =~ s!/!_-.!g;
      ## Convert any dots next to each other or at the end.
      $query =~ s/\.\./._-2e/g;
      $query =~ s/\.$/_-2e/;
      ## Convert : to _--
      $query =~ s/:/_--/g;
      ## Convert %xx to encoded form.
      $query =~ s/%([[:xdigit:]]{2})/_-$1/g;
      ## Convert % charaters to encoded form
      $query =~ s/%/_-25/g;
      ## Convert any remaining non-RFC characters.
      $query =~ s/([^a-zA-Z0-9._-])/sprintf("_-%2.2x", ord($1))/eg;
      ## Add null breaks where more than 63 in a label.
      $query =~ s/((?:\.|\G)[^.]{55}[^._]{0,4})([^.])/$1_-0.$2/g;

      return $query;
}

sub getDomain($)
{
       my ($url) = @_;

       my ($domain, $port, $path) = $url =~
                m!^(?:[^:/]*://)?([^:/\\]*)(:\d+)?(/[^?#]*)?!;
      ## Remove any trailing dots
      $domain =~ s/\.$//;
       $url = $domain;
       return $url;
}

sub getDIAKey($)
{
      ## To increase performance of lookups, only get this key once a
week
      ## and update the following line with the result.
```

```
      # return <diakey>

      my ($key) = @_;

      my $request=HTTP::Request->new('GET',
"https://subscriptions.esoft.com/api/GetRBLKey.php?OEMKey=$key");

      my $ua = LWP::UserAgent->new;
       $ua->timeout(60);
      my $response = $ua->request($request);
       if ($response->is_success) {
                  $info = $response->content;
                  $info =~ s/\015?\012//g;
                  return $info;
       } else {
                  print "Request Error:". $response->error_as_HTML .
"\n";
                  exit;
       }

}

$dns_domain = 'url1.esoft.com';
$oemkey = shift;
$serialnum = shift;
@urls = @ARGV;

if (!$serialnum || !$oemkey || !@urls ) {
      print "Invalid arguments\n";
      print "Usage: esoftlookup.pl serialnumber oemkey url\n";
      exit 1;
}
$key = getDIAKey($oemkey);

my $res = Net::DNS::Resolver->new;
#$res->debug(1);
#$res->nameservers('testrbl.esoft.com');

foreach $url (@urls) {
      ##Encode URL
      $encoded_domain = encodeURL($url);
      while( length($encoded_domain) + 4 + 7 + length($serialnum) +
length($dns_domain) >= 255 ) {
            # Remove path components then domain components until it
fits.
            next if $encoded_domain =~ s/^(.*)_-\..*$/$1/;
            $encoded_domain =~ s/^.*?\.//;
      }
      $encoded_domain =~ s/(_-\.)+$//;
      $domain = getDomain($url);

      ##Calculate CRC
```

```
	$data = lc($domain.$key.$serialnum);
	$crc = crcccitt($data);

	##Build query string
	$question =
sprintf("0.%s.%x.%s.%s",$encoded_domain,$crc,$serialnum,$dns_domain);
	print $question."\n";
	##Send the query
	my $query = $res->query($question, 'TXT');
	if($query)
	{
		foreach my $rr ($query->answer)
		{
			my @lines = split("\n", $rr->txtdata);
			foreach my $line (@lines) {
				($cats, $domain) = split ("\t", $line);
				@categories = split('_',$cats);
				print "Caching Domain: $domain\n";
				foreach $category (@categories)
				{
					print
"$category\t$categories{$category}\n";
				}
			}
		}
	} else {
		if ( $res->errorstring eq 'NXDOMAIN' ) {
			print "Unclassified\n";
		} else {
			print "Query failure: ".$res->errorstring."\n";
		}
	}
}
```

5. Direct Cloud Query Reference Code

eSoft SiteFilter OEM partners who choose to implement the Direct Cloud Query method would still greatly benefit by the use of a in-memory cache. Repeating queries for the same URL wastes bandwidth and increases latency, both of which can be reduced with the use of in-memory cache.

Creating an in-memory cache which is memory efficient, provides fast lookups performance, expires old cache entries and can be memory constrained it not a very difficult project, however, eSoft can provide the reference code to make that task much easier.

5.1. Using The Reference Code

Below is a description of each main section of the reference code. It covers DIA lookups, caching objects, cache memory limits, encoding and decoding and cleaning the cache.

NOTE: The reference software package includes a sample program named `sample.cpp` which shows in detail how the following is used.

DIA Lookup and Cache Object

The first decision to be made is picking one of the following two classes: `cached_dia_async` or `cached_dia_sync`. Both of these classes inherit from `cached_dia`. The difference is how each class implements the lookup function. The `async` class does not wait on responses before returning a category value, while the `sync` class does.

Using the `async` class means that the category values returned for the first URL lookups are likely to be uncategorized. If uncategorized URLs are allowed through the system, the effect would be that a user could load parts of a malicious, pornography or gambling page before the blocking system receives the page categories. After that further access to that URL would be blocked.

The advantage in using the `async` class is that lookup does not slow down the system and does not require multiple processes or threading.

Using the `sync` class means that the lookup will wait for a response from DIA before continuing. All requests to eSoft's data centers will have a 5 second (configurable) timeout. If it expires, the lookup function will return a value of 0 or uncategorized.

In order to use either class, you will need values for `DIA key` (see section 5.1), `serial number` and `DNS host` (provided by eSoft).

Create an object from one of these classes near the beginning of your program. Like this:

```
cached_dia_async dia(key, serial, host);
```

5.1.1. Cache Memory Limits

Choosing a memory limit will be done based on the constraints of the system. For example on a gateway appliance with minimal memory, you may need to limit total memory to 1MB while on a PC with more available memory choosing more will provide performance gains.

To set memory bounds you will set a limit value for the `memory_limiter` class. This tracks and limits how much memory the cache is using. This is done with a customized allocator passed to the STL containers that make up the cache.

Near the beginning of your program, set the limit value. Like this:

```
memory_limiter::limit(1*1024*1024);
```

5.1.2. URL Lookup and Category Decoding

Use the `cached_dia` object to look up URL categories. The lookup is invoked like this:

```
uint32_t info = dia->lookup(url);
```

Then use the `category_decoder` class to pull out the individual categories:

```
int cats[url_cache::max_categories] = {0};
generate_n(
      cats,
      url_cache::max_categories,
      category_decoder(info)
);
```

5.1.2.1. Cleaning the cache

If the cache fills up, the DIA class will automatically make room in the cache by removing the oldest entries.

eSoft's URL databases are updated continuously as old category information is replaced with newer, more accurate information. Therefore, cache entries more than a day old should be removed to make room for fresh category information. Clearing more often ensures more up to date categorization, however expiring too soon results in performance penalties. eSoft has found expiring after one day to be a good compromise, but each implementation may have different goals.

To expire old cache entries call the `clean_expired` function periodically.

The sample program does it once per hour.

NOTE: Do not call the function every time through the processing loop because it is fairly time consuming.

5.1.3. Waiting For Something To Happen

It is a good idea to keep running the UDNS event loop while waiting for things to happen. When UDNS has a DNS request outstanding, there are retransmit and query expiration timers to manage. The `cached_dia::lookup` function will run these events. But when running with `cached_dia_async`, there can be time between lookup calls when UDNS events need to run.

5.1.4. POSIX

If you are programming in a POSIX environment, you can use the `dia::wait` function like this:

```
struct pollfd pfd = {0, POLLIN, 0};
while (dia.wait(&pfd, 1, 10) >= 0 && pfd.revents == 0)
      /* do nothing */;
```

That would wait for a 10 second timeout or an input event on file descriptor 0 (standard input) while executing outstanding UDNS events. After all UDNS requests are answered or expire, `dia::wait` returns `-1`.

After that, go into a blocking read on standard input or a command socket to wait for more work.

5.1.5. Windows

Set up a timer to generate an idle event every one or two seconds. Call `dia::wait` from the idle event handler.

Data Structures

A linked list or binary tree uses pointers and too much RAM. An array takes too long to insert new entries. A `deque` is a collection of arrays that acts like one large array. It is a good choice to balance both insert time and space efficiency.

NOTE: The domain and path hash values will be 32-bit words as default, but may be adjusted smaller for more space efficiency or larger for lower collision chances

Modifying the Code

The source code is licensed for use in your applications and as an example. Modifying it or rewriting it in Java, C# or Perl is acceptable.

Some suggestions of areas you may want to look at first:

In `dia::retrieve` there is a hard-coded value of 1000 entries to clean from the cache when an insert throws `bad_alloc`.

In `cached_dia_sync::lookup` the timeout for a synchronous lookup is set to 5 seconds.

5.1.6. Adding Thread Safety

You would probably want a reader/writer lock for the cache insert, clean and lookup functions. It should be safe for any number of threads to look up values in the cache, but they should lock against writes and deletes done by the insert and clean operations.

The DIA lookup functions write into the `dia::outstanding` collection and create new entries for lookup in UDNS. They need protection by a mutex or critical section.

The `tracking_pool_allocator` class uses a `boost::fast_pool_allocator` with a `null_mutex` template argument. This might have to change for use in a threaded application.

5.1.7. Debugging

The code has some simple debugging macros defined in `dlog.h`. Change `#if 0` to `#if 1` and recompile to enable these.

5.2. Processing Overview

5.2.1. Main Loop

Check timer and Clean expired cache values if needed

Get a URL (from input or a proxy or browser plugin)

Perform a category lookup on the URL

Output the category.

5.2.2. Lookup

Get a request via lookup function call.

If in async mode: gather outstanding DIA responses into cache.

Clean expired cache values.

Check the cache for the URL, return data if found.

If in async mode:

Check list of outstanding requests (minus request filename).

If it is in the list: return uncategorized or cache hint.

Send URL to DIA (section 6.4.7).

Not in async mode:

loop until timeout:

Gather DIA responses

If URL is found: return data.

If timed out: return uncategorized or cache hint.

5.2.3. Gather Responses Into Cache

Poll all outstanding UDNS requests for results.

For each result:

Encode the returned category information into an info value.

If the recommended cache domain had a leading dot:

Insert the original request URL with a set authority bit.

Clear the authority bit on the info value.

Insert the recommended cache domain and path with the info value.

If the domain or path insertion failed because of low memory: free cache memory by freeing older entries and redo the insertion.

Do the same work for all of the recommended cache returns, but look for leading dots to determine the authority bit.

5.2.4. Clean Oldest Cache

Build a temporary binary tree (a sorted list) of pairs of domain,path indexes sorted by

timestamp.

Build a set of indexes for deletion in order of age.

Delete indexes in order, adjusting path index value for each deletion, or domain index when no paths remain.

5.2.5. Clean Expired Cache

Scan all domain and path entries for timestamps older than the cutoff time.

Delete expired path entries.

If all path entries deleted: delete domain entry.

5.2.6. Check Cache for URL

Loop on the domain, remove another subdomain on each loop:

Hash the domain.

Binary search the domain deque for the hash value.

When domain hash is found, loop on the path, remove another path component on each loop:

Hash the path.

Binary search the domain's path deque for the hash value.

When the path is found, return the information value, which contains the categories and the authority bit.

5.2.7. Send URL to DIA

Create a new string with the last part (usually a file name) trimmed off.

Check this trimmed URL against the list of outstanding DIA requests.

If found in the list, do not send and return immediately.

Create a new DIA request using UDNS.

Add the URL (minus file name) and UDNS request object to the list of outstanding DIA requests.