

ssh-ca-1.0

Generated by Doxygen 1.8.17

1 SSH CA – Backend	1
1.1 Requirements	1
1.2 Install	1
1.3 Building	1
1.4 Configure	1
1.5 Code Documentation	1
1.6 Coding Style	2
1.7 Technical Concepts	2
1.8 API Endpoints	2
2 Data Structure Index	3
2.1 Data Structures	3
3 File Index	5
3.1 File List	5
4 Data Structure Documentation	7
4.1 cgi_kvps Struct Reference	7
4.1.1 Detailed Description	8
4.2 command_arg_s Struct Reference	8
4.2.1 Detailed Description	9
4.3 command_s Struct Reference	9
4.3.1 Detailed Description	10
4.4 config_auth_s Struct Reference	10
4.4.1 Detailed Description	12
4.5 config_certify_s Struct Reference	12
4.5.1 Detailed Description	13
4.6 config_db_s Struct Reference	13
4.6.1 Detailed Description	14
4.7 config_jwt_s Struct Reference	14
4.7.1 Detailed Description	15
4.8 config_s Struct Reference	15
4.8.1 Detailed Description	16
4.9 db_s Struct Reference	16
4.9.1 Detailed Description	17
4.10 encoder_testcase_s Struct Reference	17
4.10.1 Detailed Description	18
4.11 kredis_s Struct Reference	18
4.11.1 Detailed Description	18
4.12 oidc_memory_s Struct Reference	19
4.12.1 Detailed Description	19
4.13 testable_s Struct Reference	20
4.13.1 Detailed Description	20

4.14	tmpl_t_kv_p_s Struct Reference	21
4.14.1	Detailed Description	21
4.15	user_s Struct Reference	22
4.15.1	Detailed Description	22
5	File Documentation	23
5.1	src/admin-util.c File Reference	23
5.1.1	Detailed Description	24
5.1.2	Function Documentation	24
5.1.2.1	main()	24
5.2	src/certifykey.c File Reference	25
5.2.1	Detailed Description	25
5.2.2	Function Documentation	26
5.2.2.1	main()	26
5.3	src/cgi.c File Reference	26
5.3.1	Detailed Description	27
5.3.2	Function Documentation	27
5.3.2.1	cgi_kv_p_add()	28
5.3.2.2	cgi_kv_p_add_delimbased()	29
5.3.2.3	cgi_kv_p_create()	30
5.3.2.4	cgi_kv_p_free()	31
5.3.2.5	cgi_kv_p_get()	32
5.3.2.6	cgi_parse_http_cookie()	33
5.3.2.7	cgi_parse_query_string()	34
5.3.2.8	cgi_testenv()	35
5.4	src/cgi.h File Reference	36
5.4.1	Detailed Description	37
5.4.2	Function Documentation	37
5.4.2.1	cgi_kv_p_add()	37
5.4.2.2	cgi_kv_p_add_delimbased()	39
5.4.2.3	cgi_kv_p_create()	40
5.4.2.4	cgi_kv_p_free()	41
5.4.2.5	cgi_kv_p_get()	42
5.4.2.6	cgi_parse_http_cookie()	44
5.4.2.7	cgi_parse_query_string()	45
5.4.2.8	cgi_testenv()	45
5.5	src/common.c File Reference	46
5.5.1	Detailed Description	48
5.5.2	Function Documentation	48
5.5.2.1	base64_decode()	48
5.5.2.2	base64_encode()	49
5.5.2.3	base64url_decode()	49

5.5.2.4	base64url_encode()	50
5.5.2.5	filetojson()	52
5.5.2.6	filetostr()	53
5.5.2.7	get_bool_from_json()	53
5.5.2.8	get_number_from_json()	54
5.5.2.9	get_string_from_json()	55
5.5.2.10	hex_to_int()	56
5.5.2.11	hmac_sha256()	56
5.5.2.12	int_to_hex()	57
5.5.2.13	is_percent_encoded()	58
5.5.2.14	is_valid_hex()	59
5.5.2.15	is_valid_uuid_v4()	59
5.5.2.16	isprefix()	60
5.5.2.17	percent_decode()	61
5.5.2.18	percent_encode()	62
5.5.2.19	remove_whitespace()	63
5.5.2.20	str_append()	64
5.5.2.21	uuid_v4_gen()	65
5.5.2.22	writestreamtofile()	66
5.6	src/common.h File Reference	66
5.6.1	Detailed Description	69
5.6.2	Function Documentation	69
5.6.2.1	base64_decode()	69
5.6.2.2	base64_encode()	70
5.6.2.3	base64url_decode()	70
5.6.2.4	base64url_encode()	71
5.6.2.5	filetojson()	73
5.6.2.6	filetostr()	74
5.6.2.7	get_bool_from_json()	74
5.6.2.8	get_number_from_json()	75
5.6.2.9	get_string_from_json()	76
5.6.2.10	hex_to_int()	77
5.6.2.11	hmac_sha256()	77
5.6.2.12	int_to_hex()	78
5.6.2.13	is_percent_encoded()	79
5.6.2.14	is_valid_hex()	80
5.6.2.15	is_valid_uuid_v4()	80
5.6.2.16	isprefix()	81
5.6.2.17	percent_decode()	82
5.6.2.18	percent_encode()	83
5.6.2.19	remove_whitespace()	84
5.6.2.20	str_append()	85

5.6.2.21	uuid_v4_gen()	86
5.6.2.22	writestreamtofile()	87
5.7	src/config.c File Reference	87
5.7.1	Detailed Description	88
5.7.1.1	config.json Layout	88
5.7.2	Function Documentation	89
5.7.2.1	config_free()	89
5.7.2.2	config_parse()	89
5.8	src/config.h File Reference	90
5.8.1	Detailed Description	91
5.8.2	Function Documentation	91
5.8.2.1	config_free()	91
5.8.2.2	config_parse()	92
5.9	src/db.c File Reference	92
5.9.1	Detailed Description	93
5.9.1.1	Database Layout	93
5.9.2	Function Documentation	94
5.9.2.1	db_close()	94
5.9.2.2	db_connect()	95
5.9.2.3	db_jwtrefresh_token_create()	95
5.9.2.4	db_jwtrefresh_token_refresh()	97
5.9.2.5	db_user_get()	98
5.9.2.6	db_user_store()	100
5.9.2.7	db_users_get_all()	101
5.10	src/db.h File Reference	102
5.10.1	Detailed Description	104
5.10.2	Macro Definition Documentation	104
5.10.2.1	DB_PREFIX_USERS	104
5.10.3	Function Documentation	104
5.10.3.1	db_close()	104
5.10.3.2	db_connect()	105
5.10.3.3	db_jwtrefresh_token_create()	106
5.10.3.4	db_jwtrefresh_token_refresh()	108
5.10.3.5	db_user_get()	109
5.10.3.6	db_user_store()	111
5.10.3.7	db_users_get_all()	112
5.11	src/jwt.c File Reference	113
5.11.1	Detailed Description	114
5.11.2	Function Documentation	114
5.11.2.1	jwt_decode()	115
5.11.2.2	jwt_encode_and_sign()	116
5.11.2.3	jwt_gen_payload()	117

5.11.2.4	jwt_has_expired()	118
5.11.2.5	jwt_simple_decode()	119
5.11.2.6	jwt_simple_encode()	120
5.11.2.7	jwt_verify_encoding()	121
5.11.2.8	jwt_verify_signature()	122
5.12	src/jwt.h File Reference	123
5.12.1	Detailed Description	124
5.12.2	Function Documentation	124
5.12.2.1	jwt_decode()	124
5.12.2.2	jwt_encode_and_sign()	126
5.12.2.3	jwt_gen_payload()	127
5.12.2.4	jwt_has_expired()	128
5.12.2.5	jwt_simple_decode()	129
5.12.2.6	jwt_simple_encode()	130
5.12.2.7	jwt_verify_encoding()	131
5.12.2.8	jwt_verify_signature()	132
5.13	src/kredis.c File Reference	133
5.13.1	Detailed Description	134
5.13.2	Function Documentation	135
5.13.2.1	kredis_connect()	135
5.13.2.2	kredis_del()	136
5.13.2.3	kredis_error()	137
5.13.2.4	kredis_error_string()	138
5.13.2.5	kredis_free()	139
5.13.2.6	kredis_get()	140
5.13.2.7	kredis_hash_exists()	141
5.13.2.8	kredis_hash_get()	141
5.13.2.9	kredis_hash_set()	142
5.13.2.10	kredis_ping()	143
5.13.2.11	kredis_set()	144
5.13.2.12	kredis_set_add()	145
5.13.2.13	kredis_set_free()	146
5.13.2.14	kredis_set_get()	147
5.13.2.15	kredis_set_ismember()	148
5.13.2.16	kredis_set_num()	149
5.13.2.17	kredis_set_rem()	149
5.13.2.18	kredis_set_with_ex()	150
5.14	src/kredis.h File Reference	151
5.14.1	Detailed Description	153
5.14.2	Function Documentation	153
5.14.2.1	kredis_connect()	153
5.14.2.2	kredis_del()	154

5.14.2.3	kredis_error()	155
5.14.2.4	kredis_error_string()	156
5.14.2.5	kredis_free()	157
5.14.2.6	kredis_get()	158
5.14.2.7	kredis_hash_exists()	159
5.14.2.8	kredis_hash_get()	160
5.14.2.9	kredis_hash_set()	161
5.14.2.10	kredis_ping()	161
5.14.2.11	kredis_set()	162
5.14.2.12	kredis_set_add()	163
5.14.2.13	kredis_set_free()	164
5.14.2.14	kredis_set_get()	165
5.14.2.15	kredis_set_ismember()	166
5.14.2.16	kredis_set_num()	167
5.14.2.17	kredis_set_rem()	168
5.14.2.18	kredis_set_with_ex()	169
5.15	src/oauth.c File Reference	170
5.15.1	Detailed Description	170
5.15.2	Function Documentation	170
5.15.2.1	oauth_gen_code_challenge()	171
5.15.2.2	oauth_gen_code_verifier()	172
5.15.2.3	oauth_gen_state()	172
5.15.2.4	oauth_gen_url()	173
5.15.2.5	oauth_verify_state_encoding()	174
5.16	src/oauth.h File Reference	175
5.16.1	Detailed Description	176
5.16.2	Function Documentation	176
5.16.2.1	oauth_gen_code_challenge()	176
5.16.2.2	oauth_gen_code_verifier()	177
5.16.2.3	oauth_gen_state()	178
5.16.2.4	oauth_gen_url()	179
5.16.2.5	oauth_verify_state_encoding()	179
5.17	src/oauthcallback.c File Reference	180
5.17.1	Detailed Description	181
5.17.2	Function Documentation	181
5.17.2.1	main()	181
5.18	src/oidc.c File Reference	182
5.18.1	Detailed Description	183
5.18.2	Function Documentation	183
5.18.2.1	oidc_get_external_subject_id_from_identity_provider()	183
5.19	src/oidc.h File Reference	184
5.19.1	Detailed Description	185

5.19.2 Function Documentation	185
5.19.2.1 oidc_get_external_subject_id_from_identity_provider()	185
5.20 src/refreshlogin.c File Reference	186
5.20.1 Detailed Description	187
5.20.2 Function Documentation	187
5.20.2.1 main()	187
5.21 src/requestoauthurl.c File Reference	188
5.21.1 Detailed Description	188
5.21.2 Function Documentation	189
5.21.2.1 main()	189
5.22 src/template_engine.c File Reference	189
5.22.1 Detailed Description	190
5.22.1.1 TEMPLATING LANGUAGE	190
5.22.2 Function Documentation	190
5.22.2.1 tmpl_render()	190
5.22.2.2 tmpl_render_from_file()	192
5.23 src/template_engine.h File Reference	192
5.23.1 Detailed Description	194
5.23.2 Function Documentation	194
5.23.2.1 tmpl_render()	194
5.23.2.2 tmpl_render_from_file()	195
5.24 src/test-util.c File Reference	196
5.24.1 Detailed Description	197
5.24.2 Function Documentation	197
5.24.2.1 main()	197
5.25 src/user.c File Reference	198
5.25.1 Detailed Description	199
5.25.2 Function Documentation	199
5.25.2.1 user_free()	199
5.25.2.2 user_get()	199
5.25.2.3 user_get_by_oidc_external_subject_id()	201
5.25.2.4 user_parse()	202
5.26 src/user.h File Reference	203
5.26.1 Detailed Description	204
5.26.2 Function Documentation	204
5.26.2.1 user_free()	204
5.26.2.2 user_get()	205
5.26.2.3 user_get_by_oidc_external_subject_id()	206
5.26.2.4 user_parse()	207

Chapter 1

SSH CA – Backend

1.1 Requirements

```
sudo apt install libcurl4-openssl-dev build-essential libssl-dev git
sudo apt install nginx fcgiwrap
sudo apt install doxygen doxygen-latex graphviz
```

- A Redis Server. (Version: $\geq 2.6.12$)

1.2 Install

1. Go to /opt/ and clone this repo `git clone --recurse-submodules git@gitlab.gwdg.de↵:asta/dnd-referat/ssh-ca.git.`
2. Comment out all location blocks in your nginx site config and add this line: `include /opt/ssh-ca/site.conf;`
3. `sudo service nginx restart`

Tested on GWDG Cloud Servers Ubuntu 20.04 LTS.

1.3 Building

TODO (make all)

1.4 Configure

TODO

1.5 Code Documentation

The full backend/code documentation can be found [here as HTML](#) and [here as PDF](#).

1.6 Coding Style

TODO

1.7 Technical Concepts

TODO

1.8 API Endpoints

TODO

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

cgi_kv_p_s	Key-value-pairs struct	7
command_arg_s	Command argument struct	8
command_s	Command struct	9
config_auth_s	Native representation of config.auth	10
config_certify_s	Native representation of config["ssh-sign"]	12
config_db_s	Native representation of config.db	13
config_jwt_s	Native representation of config.jwt	14
config_s	Config struct	15
db_s	Database struct	16
encoder_testcase_s	Struct used for testing encoders	17
kredis_s	18
oidc_memory_s	Internal struct used for receiving curl callback data	19
testable_s	Testcase struct	20
tmpl_t_kv_p_s	Template Key-Value-Pair Struct	21
user_s	User struct	22

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

src/admin-util.c	Simple tool administration for administrating the SSH CA	23
src/certifykey.c	Key signing utility. handles endpoint at /cgi/certifykey	25
src/cgi.c	Code for working with CGI	26
src/cgi.h	Header file for cgi.c	36
src/common.c	A bunch of useful/common code	46
src/common.h	Header file for common.c	66
src/config.c	Code for parsing the config into a native format	87
src/config.h	Header file for config.c	90
src/db.c	Code for interacting with the database on a 'higher'-level	92
src/db.h	Header file for db.c	102
src/jwt.c	JSON Web Token	113
src/jwt.h	Header file for jwt.c	123
src/kredis.c	Simple ABI for hiredis	133
src/kredis.h	Header file for kredis.c	151
src/oauth.c	Useful code for OAuth Authentication	170
src/oauth.h	Header file for oauth.c	175
src/oauthcallback.c	OAuth callback receiver. handles endpoint at /cgi/oauthcallback	180
src/oidc.c	Open ID Connect implementation	182

src/oidc.h	
Header file for oidc.c	184
src/refreshlogin.c	
Utility for refreshing the jwtrefresh token and getting a JWT. handles endpoint at /cgi/refreshlogin	186
src/requestoathurl.c	
Oauth url generator. handles endpoint at /cgi/requestoathurl	188
src/template_engine.c	
Template rendering code	189
src/template_engine.h	
Header file for template_engine.c	192
src/test-util.c	
Tool for testing PARTs of this program	196
src/user.c	
User related code	198
src/user.h	
Header file for user.c	203

Chapter 4

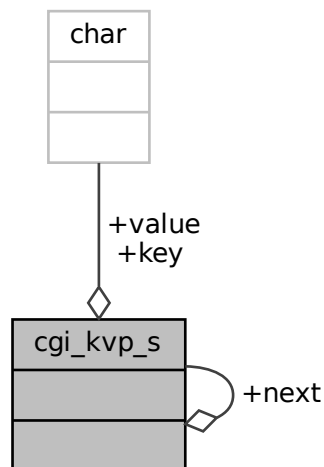
Data Structure Documentation

4.1 cgi_kvps Struct Reference

key-value-pairs struct

```
#include <cgi.h>
```

Collaboration diagram for cgi_kvps:



Data Fields

- char * [key](#)
the key
- char * [value](#)
the value associated with the key
- struct [cgi_kvps](#) * [next](#)
the next pair

4.1.1 Detailed Description

key-value-pairs struct

Basically a linked list of key-value-pairs.

Definition at line 22 of file cgi.h.

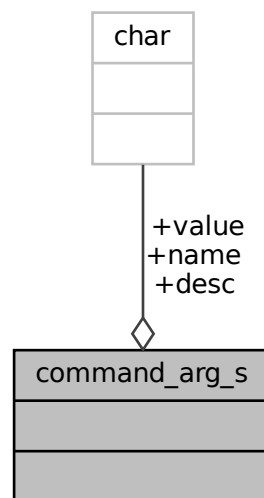
The documentation for this struct was generated from the following file:

- [src/cgi.h](#)

4.2 `command_arg_s` Struct Reference

command argument struct

Collaboration diagram for `command_arg_s`:



Data Fields

- `char name` [32]
name of the argument
- `char desc` [128]
description of the argument
- `char value` [MAXARGLEN+1]
buffer for storing the value

4.2.1 Detailed Description

command argument struct

Definition at line 44 of file admin-util.c.

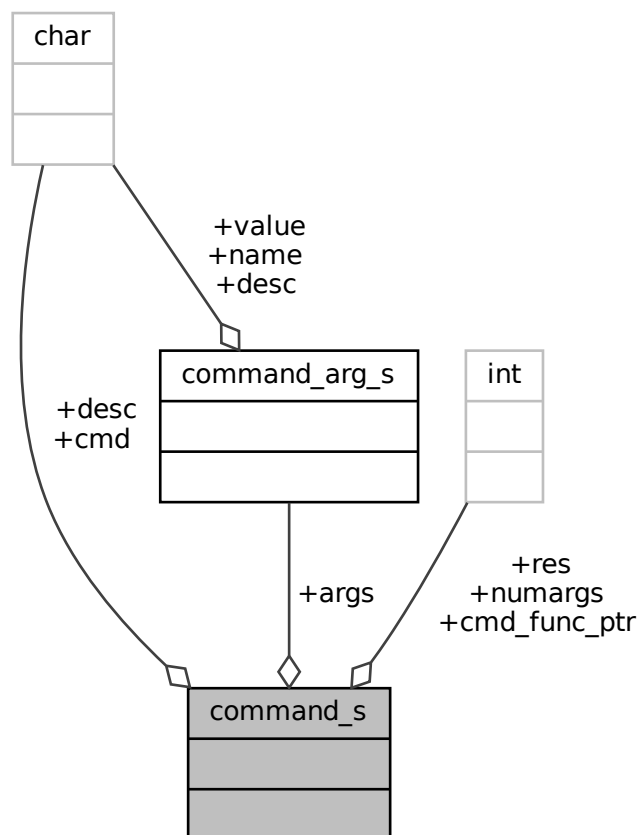
The documentation for this struct was generated from the following file:

- [src/admin-util.c](#)

4.3 command_s Struct Reference

command struct

Collaboration diagram for command_s:



Data Fields

- char `cmd` [32]
name of the command
- char `desc` [128]
description of the command
- int `numargs`
number of arguments the command expects
- struct `command_arg_s args` [4]
declaration of arguments
- int(* `cmd_func_ptr`)(char *errmsg, const char *cmd, const struct `config_s` *config, `db_t` *db, struct `command_arg_s` *args, int numargs)
command function
- int `res`
return value of command function

4.3.1 Detailed Description

command struct

Definition at line 54 of file admin-util.c.

The documentation for this struct was generated from the following file:

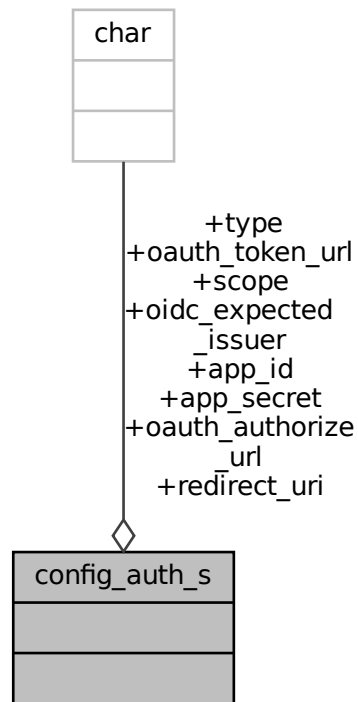
- [src/admin-util.c](#)

4.4 config_auth_s Struct Reference

native representation of config.auth

```
#include <config.h>
```

Collaboration diagram for config_auth_s:



Data Fields

- `char * type`
authentication type (currently only oidc (OpenID Connect) is supported)
- `char * oauth_authorize_url`
the oauth authorize endpoint of the identity provider
- `char * app_id`
oauth app id
- `char * redirect_uri`
oauth redirect url
- `char * scope`
oauth scope
- `char * app_secret`
oauth scope
- `char * oauth_token_url`
the oauth/oidc token endpoint of the identity provider
- `char * oidc_expected_issuer`
the expected issuer (iss) claim in id_token

4.4.1 Detailed Description

native representation of config.auth

Definition at line 32 of file config.h.

The documentation for this struct was generated from the following file:

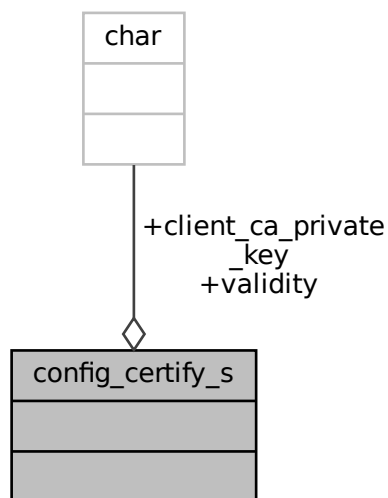
- [src/config.h](#)

4.5 config_certify_s Struct Reference

native representation of config["ssh-sign"]

```
#include <config.h>
```

Collaboration diagram for config_certify_s:



Data Fields

- `char * client_ca_private_key`
path on the server to the private client-ca key
- `char * validity`
how long a certified key should be valid for, see TIME FORMATS in sshd_config(5)

4.5.1 Detailed Description

native representation of config["ssh-sign"]

Definition at line 57 of file config.h.

The documentation for this struct was generated from the following file:

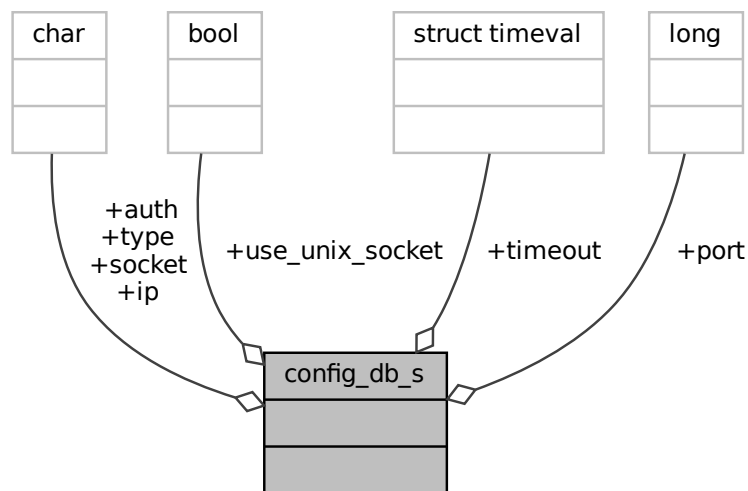
- [src/config.h](#)

4.6 config_db_s Struct Reference

native representation of config.db

```
#include <config.h>
```

Collaboration diagram for config_db_s:



Data Fields

- char * [type](#)
database type (currently only redis supported)
- char * [ip](#)
server ip
- long [port](#)
server port
- char * [socket](#)
server unix socket
- bool [use_unix_socket](#)
whether to use the unix socket or ip+port
- struct timeval [timeout](#)
redis timeout
- char * [auth](#)
redis auth string, may be NULL if no authentication

4.6.1 Detailed Description

native representation of config.db

Definition at line 18 of file config.h.

The documentation for this struct was generated from the following file:

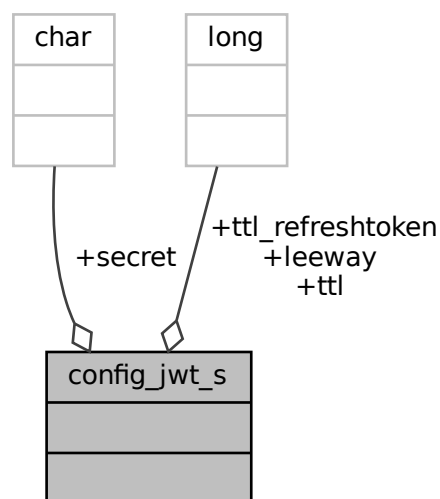
- [src/config.h](#)

4.7 config_jwt_s Struct Reference

native representation of config.jwt

```
#include <config.h>
```

Collaboration diagram for config_jwt_s:



Data Fields

- `char * secret`
JWT secret.
- `long ttl`
seconds until a JWT expires
- `long leeway`
leeway to be added to ttl
- `long ttl_refreshtoken`
seconds until a refreshtoken expires

4.7.1 Detailed Description

native representation of config.jwt

Definition at line 46 of file config.h.

The documentation for this struct was generated from the following file:

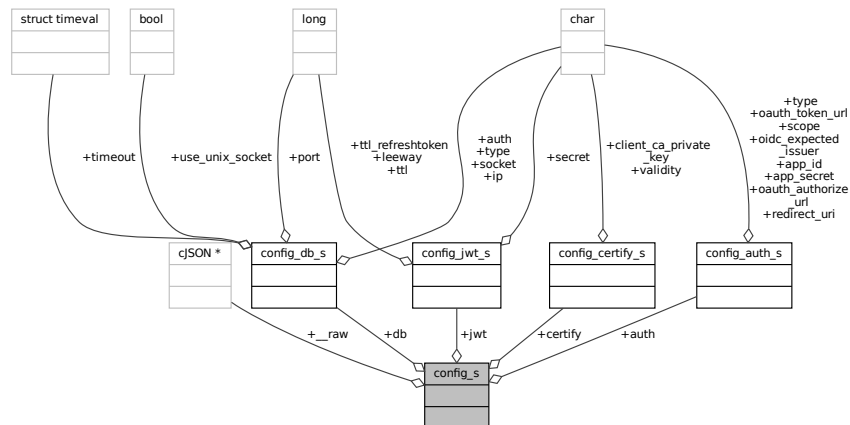
- [src/config.h](#)

4.8 config_s Struct Reference

config struct

```
#include <config.h>
```

Collaboration diagram for config_s:



Data Fields

- cJSON * [__raw](#)
the raw json data (internal use only!)
- struct [config_db_s db](#)
config.db
- struct [config_auth_s auth](#)
config.auth
- struct [config_jwt_s jwt](#)
config.jwt
- struct [config_certify_s certify](#)
config["ssh-sign"]

4.8.1 Detailed Description

config struct

Definition at line 66 of file config.h.

The documentation for this struct was generated from the following file:

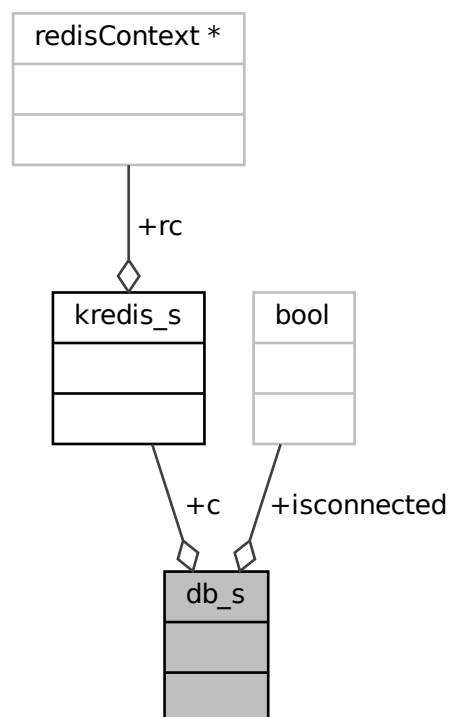
- [src/config.h](#)

4.9 db_s Struct Reference

database struct

```
#include <db.h>
```

Collaboration diagram for db_s:



Data Fields

- `bool isconnected`
whether the kredis context is connected to a database
- `kredis_t * c`
the kredis context

4.9.1 Detailed Description

database struct

Definition at line 42 of file db.h.

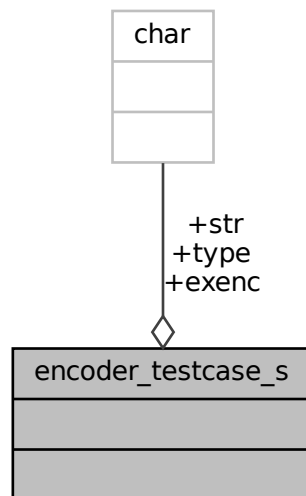
The documentation for this struct was generated from the following file:

- [src/db.h](#)

4.10 encoder_testcase_s Struct Reference

struct used for testing encoders

Collaboration diagram for encoder_testcase_s:



Data Fields

- `char type`
'+' : str is valid '-' : str is invalid '\0' : end of array
- `char str [256]`
testcase string
- `char exenc [256 *3+1]`
expected encoded of str

4.10.1 Detailed Description

struct used for testing encoders

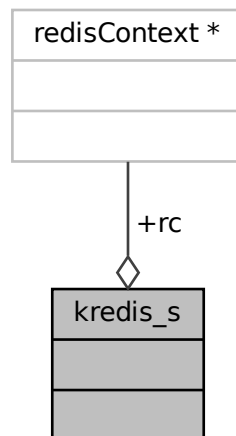
Definition at line 46 of file test-util.c.

The documentation for this struct was generated from the following file:

- [src/test-util.c](#)

4.11 kredis_s Struct Reference

Collaboration diagram for kredis_s:



Data Fields

- `redisContext * rc`
the redisContext from hiredis

4.11.1 Detailed Description

A kredis context struct

Definition at line 28 of file kredis.c.

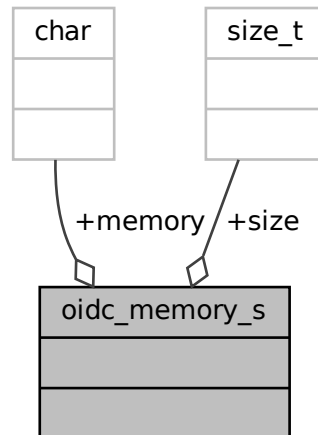
The documentation for this struct was generated from the following file:

- [src/kredis.c](#)

4.12 oidc_memory_s Struct Reference

internal struct used for receiving curl callback data

Collaboration diagram for oidc_memory_s:



Data Fields

- `char * memory`
pointer to the allocated memory, containing the data
- `size_t size`
size of memory

4.12.1 Detailed Description

internal struct used for receiving curl callback data

Definition at line 30 of file `oidc.c`.

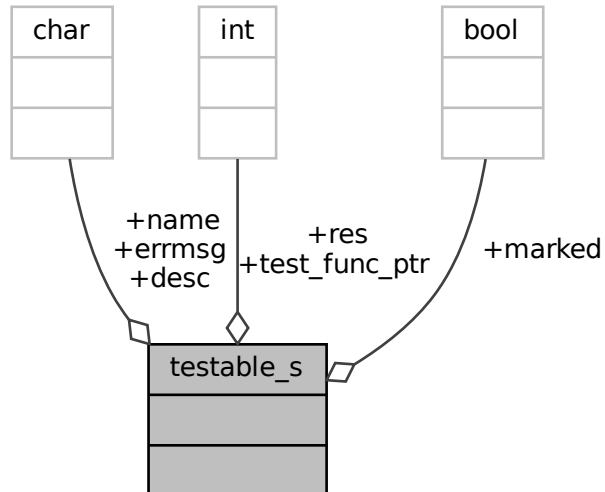
The documentation for this struct was generated from the following file:

- `src/oidc.c`

4.13 testable_s Struct Reference

testcase struct

Collaboration diagram for testable_s:



Data Fields

- char [name](#) [32]
name of the test
- char [desc](#) [128]
description of the test
- int(* [test_func_ptr](#))(char *[errmsg](#), const char *[name](#), const struct [config_s](#) *[config](#))
test function
- bool [marked](#)
true: should be tested false: should not be tested
- int [res](#)
return value of test function. 0 indicated failure
- char [errmsg](#) [ERRMSGMAXSIZE]
buffer for error message

4.13.1 Detailed Description

testcase struct

Definition at line 35 of file test-util.c.

The documentation for this struct was generated from the following file:

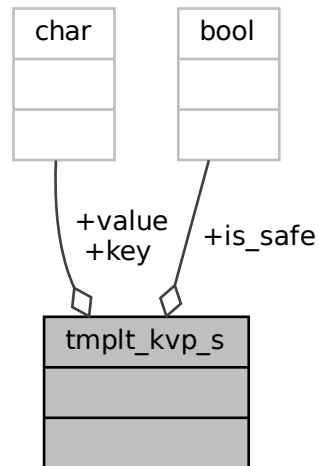
- [src/test-util.c](#)

4.14 tmplt_kvp_s Struct Reference

Template Key-Value-Pair Struct.

```
#include <template_engine.h>
```

Collaboration diagram for tmplt_kvp_s:



Data Fields

- const char * [key](#)
the key
- const char * [value](#)
the value that will replace %key%%
- bool [is_safe](#)
whether the string is safe and can be used as is, or the string is unsafe and needs to be encoded

4.14.1 Detailed Description

Template Key-Value-Pair Struct.

Note

An Array of [tmplt_kvp_s](#) should be terminated by an element with key set to NULL.

Definition at line 33 of file [template_engine.h](#).

The documentation for this struct was generated from the following file:

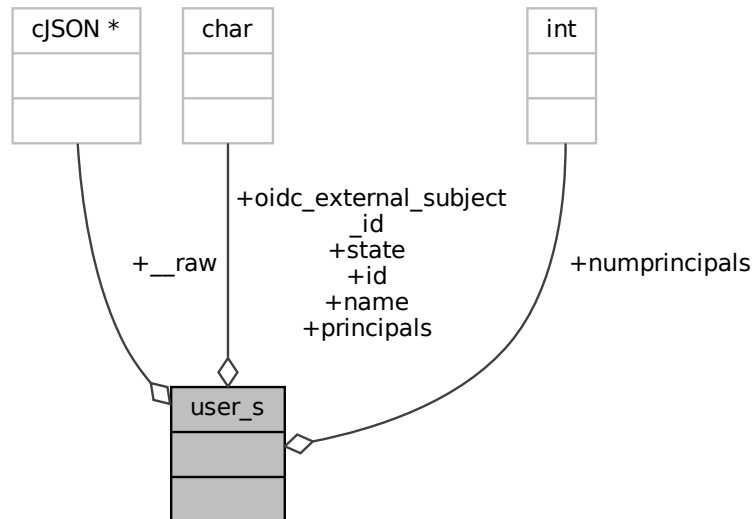
- [src/template_engine.h](#)

4.15 user_s Struct Reference

user struct

```
#include <user.h>
```

Collaboration diagram for user_s:



Data Fields

- cJSON * [__raw](#)
the raw json data (internal use only!)
- char * [id](#)
the userid
- char * [name](#)
the name
- char * [oidc_external_subject_id](#)
the users external subject id in the identity provider
- char * [state](#)
the state, either "active" or "blocked"
- int [numprincipals](#)
the number of principals in principals
- char ** [principals](#)
null-terminated list of principal strings

4.15.1 Detailed Description

user struct

Definition at line 18 of file user.h.

The documentation for this struct was generated from the following file:

- [src/user.h](#)

Chapter 5

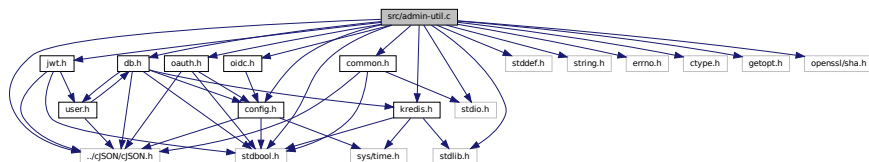
File Documentation

5.1 src/admin-util.c File Reference

Simple tool administration for administrating the SSH CA.

```
#include "common.h"  
#include "db.h"  
#include "oidc.h"  
#include "oauth.h"  
#include "kredis.h"  
#include "jwt.h"  
#include "config.h"  
#include "../cJSON/cJSON.h"  
#include <stdio.h>  
#include <stdlib.h>  
#include <stdbool.h>  
#include <string.h>  
#include <errno.h>  
#include <ctype.h>  
#include <getopt.h>  
#include <openssl/sha.h>
```

Include dependency graph for admin-util.c:



Data Structures

- struct `command_arg_s`
command argument struct
- struct `command_s`
command struct

Macros

- #define `MAXFILEPATHLEN` 256
the maximum length of a file path
- #define `MAXARGLEN` 256
the maximum length of a command argument

Functions

- int `main` (int argc, char **argv, char **envp)
main function for admin-util

5.1.1 Detailed Description

Simple tool administration for administrating the SSH CA.

Note

This tool will be removed as soon as a fully working REST API + Frontend exists.

See also

<https://gitlab.gwdg.de/asta/dnd-referat/ssh-ca/-/issues/14>

5.1.2 Function Documentation

5.1.2.1 main()

```
int main (  
    int argc,  
    char ** argv,  
    char ** envp )
```

main function for admin-util

Parameters

in	<code>argc</code>	number of arguments from the command line
in	<code>argv</code>	arguments from the command line
in	<code>envp</code>	environment variables

Returns

the exit code of this program.

Return values

0	success
!0	failure

Definition at line 422 of file admin-util.c.

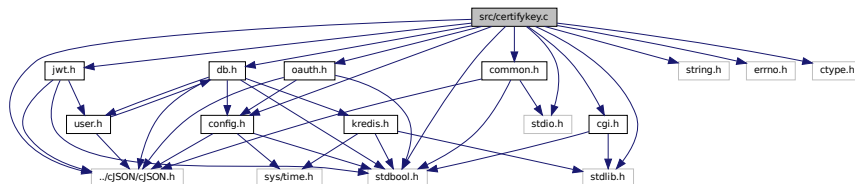
References `config_s::db`, `ERRMSGMAXSIZE`, `MAXFILEPATHLEN`, and `command_s::res`.

5.2 src/certifykey.c File Reference

key signing utility. handles endpoint at `/cgi/certifykey`

```
#include "common.h"
#include "cgi.h"
#include "db.h"
#include "jwt.h"
#include "oauth.h"
#include "config.h"
#include "../cJSON/cJSON.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
```

Include dependency graph for `certifykey.c`:



Functions

- `int main` (void)
main function for certifykey

5.2.1 Detailed Description

key signing utility. handles endpoint at `/cgi/certifykey`

Request: Expects a POST Request with Content-Type set to `text/plain`. The Request Header must contain a valid Authorization Bearer Token (JWT). The User must be 'active'. The Body must contain the PublicKey the User wants to get signed.

Response (if successful): Content-Type: `text/plain` The Body contains the signed PublicKey of the User.

5.2.2 Function Documentation

5.2.2.1 main()

```
int main (  
        void )
```

main function for certifykey

Returns

the exit code of this program.

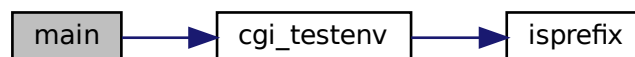
Return values

0	success
!0	failure

Definition at line 174 of file certifykey.c.

References `cgi_testenv()`, `config_s::db`, `ERRMSGMAXSIZE`, and `UIDLEN`.

Here is the call graph for this function:



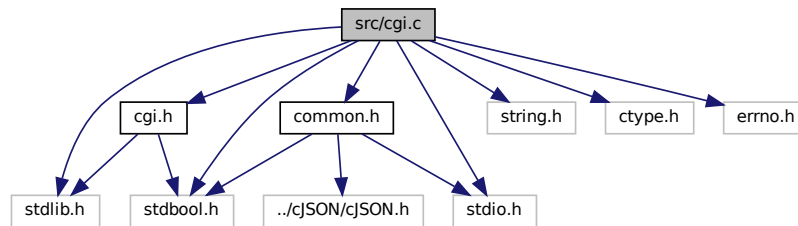
5.3 src/cgi.c File Reference

Code for working with CGI.

```
#include "cgi.h"  
#include "common.h"  
#include <stdio.h>  
#include <stdlib.h>  
#include <stdbool.h>  
#include <string.h>  
#include <ctype.h>
```

```
#include <errno.h>
```

Include dependency graph for cgi.c:



Functions

- struct `cgi_kvp_s` * `cgi_parse_query_string` (const char *qryst)

Parses a QUERY_STRING.
- struct `cgi_kvp_s` * `cgi_parse_http_cookie` (const char *cokstr)

Parses a HTTP_COOKIE string.
- struct `cgi_kvp_s` * `cgi_kvp_create` (void)

Create a key-value-pairs list.
- void `cgi_kvp_free` (struct `cgi_kvp_s` *kvp)

Frees a key-value-pairs list.
- bool `cgi_kvp_add` (struct `cgi_kvp_s` *kvp, const char *key, const char *value)

Adds a key-value-pair to a key-value-pairs list.
- bool `cgi_kvp_add_delimbased` (struct `cgi_kvp_s` *kvp, char *keyvalue, const char delim)

Adds a key-value-pair to a key-value-pairs list but the key-value-pair is represented as a single string seperated by a delimiter.
- char * `cgi_kvp_get` (struct `cgi_kvp_s` *kvp, const char *key)

Gets a value associated with a given key from a key-value-pairs list.
- int `cgi_testenv` (char *errmsg, char method, const char *post_content_type)
- void `cgi_printstatuscodeasstr` (int statuscode)
- void `cgi_printerror` (char *errmsg, int statuscode)

5.3.1 Detailed Description

Code for working with CGI.

Note

This file may get replaced in future by the REST API Code.

5.3.2 Function Documentation

5.3.2.1 cgi_kvp_add()

```
bool cgi_kvp_add (
    struct cgi_kvp_s * kvp,
    const char * key,
    const char * value )
```

Adds a key-value-pair to a key-value-pairs list.

Note

Neither key or value can be NULL.

Duplicate keys can exist and `cgi_kvp_get()` will only retrieve the first one. Thus you cannot override a value.

Parameters

in, out	<i>kvp</i>	the key-value-pairs list
in	<i>key</i>	the key
in	<i>value</i>	the value

Returns

whether the pair was successfully added to the list

Return values

<i>true</i>	success
<i>false</i>	an error occured and errno might be set

Definition at line 179 of file cgi.c.

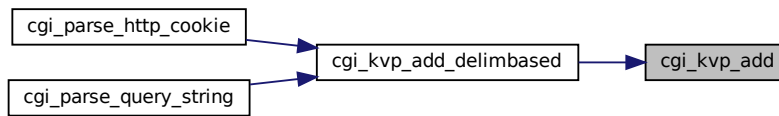
References `cgi_kvp_create()`, `cgi_kvp_s::key`, `cgi_kvp_s::next`, and `cgi_kvp_s::value`.

Referenced by `cgi_kvp_add_delimbased()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.2 `cgi_kvp_add_delimbased()`

```

bool cgi_kvp_add_delimbased (
    struct cgi_kvp_s * kvp,
    char * keyvalue,
    const char delim )
  
```

Adds a key-value-pair to a key-value-pairs list but the key-value-pair is represented as a single string seperated by a delimiter.

If the delimiter is not or more than once in keyvalue, then 0 is returned.

This function is destructive and will replace the delimiter in keyvalue with '\0'.

The delimiter cannot be '\0'.

See also

[cgi_kvp_add\(\)](#)

Parameters

<i>in, out</i>	<i>kvp</i>	the key-value-pairs list
<i>in</i>	<i>keyvalue</i>	the key and value in the same string seperated by the delimiter
<i>in</i>	<i>delim</i>	the delimiter

Returns

whether the pair was successfully added to the list

Return values

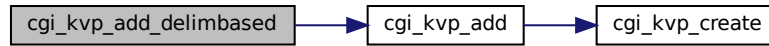
<i>true</i>	success
<i>false</i>	an error ocured and errno might be set or the delimiter was not exactly once in the string

Definition at line 235 of file `cgi.c`.

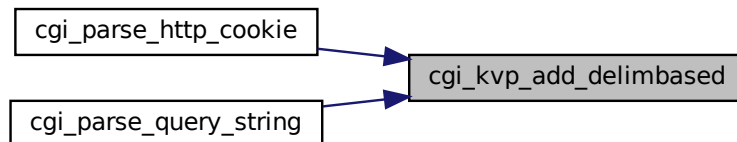
References `cgi_kvp_add()`.

Referenced by `cgi_parse_http_cookie()`, and `cgi_parse_query_string()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.3 `cgi_kvp_create()`

```
struct cgi_kvp_s* cgi_kvp_create (
    void )
```

Create a key-value-pairs list.

Returns

a key-value-pairs struct

Return values

<code>NULL</code>	an error occurred and <code>errno</code> is set
-------------------	---

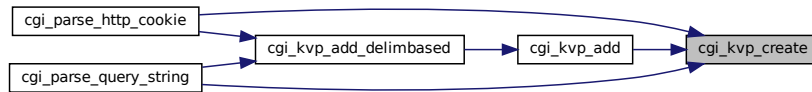
The returned pointer must be freed using `cgi_kvp_free()`.

Definition at line 131 of file `cgi.c`.

References `cgi_kvp_s::key`, `cgi_kvp_s::next`, and `cgi_kvp_s::value`.

Referenced by `cgi_kvp_add()`, `cgi_parse_http_cookie()`, and `cgi_parse_query_string()`.

Here is the caller graph for this function:



5.3.2.4 `cgi_kvp_free()`

```
void cgi_kvp_free (
    struct cgi_kvp_s * kvp )
```

Frees a key-value-pairs list.

Parameters

<code>in, out</code>	<code>kvp</code>	the key-value-pairs list
----------------------	------------------	--------------------------

See also

[cgi_kvp_create\(\)](#)

Definition at line 152 of file `cgi.c`.

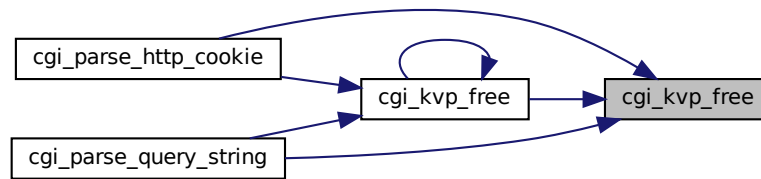
References `cgi_kvp_free()`, `cgi_kvp_s::key`, `cgi_kvp_s::next`, and `cgi_kvp_s::value`.

Referenced by `cgi_kvp_free()`, `cgi_parse_http_cookie()`, and `cgi_parse_query_string()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.5 cgi_kvp_get()

```
char* cgi_kvp_get (
    struct cgi_kvp_s * kvp,
    const char * key )
```

Gets a value associated with a given key from a key-value-pairs list.

Parameters

in, out	<i>kvp</i>	the key-value-pairs list
in	<i>key</i>	the key

Returns

the value associated with the key

Return values

<i>NULL</i>	no value is associated with the key
-------------	-------------------------------------

Definition at line 262 of file cgi.c.

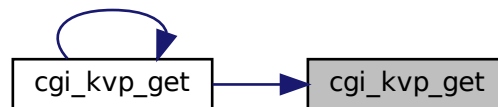
References `cgi_kvp_get()`, `cgi_kvp_s::key`, `cgi_kvp_s::next`, and `cgi_kvp_s::value`.

Referenced by `cgi_kvp_get()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.6 cgi_parse_http_cookie()

```

struct cgi_kvp_s* cgi_parse_http_cookie (
    const char * cokstr )
  
```

Parses a HTTP_COOKIE string.

This is not standard conforming!

The HTTP_COOKIE sting is a a list of key-value-pairs seperated by ';' characters. Each key-value-pair is intern seperated by an '=' character. Whitespaces are ignored.

Parameters

in	<i>cokstr</i>	the HTTP_COOKIE string.
----	---------------	-------------------------

Returns

a key-value-pairs list containing the key-value-pairs

Return values

<i>NULL</i>	an error occurred and errno might be set. maybe the HTTP_COOKIE string is invalid?
-------------	--

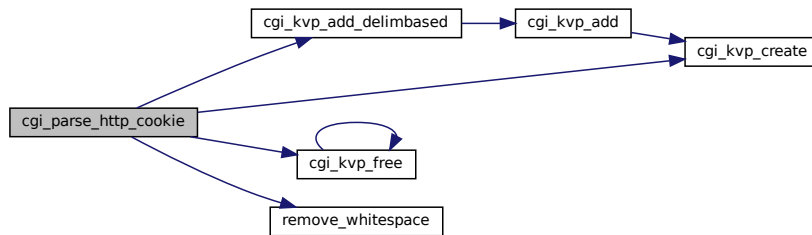
The values can be accessed via [cgi_kvp_get\(\)](#).

The returned pointer must be freed using [cgi_kvp_free\(\)](#).

Definition at line 86 of file cgi.c.

References [cgi_kvp_add_delimbased\(\)](#), [cgi_kvp_create\(\)](#), [cgi_kvp_free\(\)](#), and [remove_whitespace\(\)](#).

Here is the call graph for this function:



5.3.2.7 cgi_parse_query_string()

```
struct cgi_kvp_s* cgi_parse_query_string (
    const char * qrystr )
```

Parses a QUERY_STRING.

This is not standard conforming!

The QUERY_STRING is a a list of key-value-pairs separated by '&' characters. Each key-value-pair is intern separated by an '=' character.

Parameters

in	<i>qrystr</i>	the QUERY_STRING
----	---------------	------------------

Returns

a key-value-pairs list containing the key-value-pairs

Return values

<i>NULL</i>	an error occurred and errno might be set. maybe the QUERY_STRING is invalid?
-------------	--

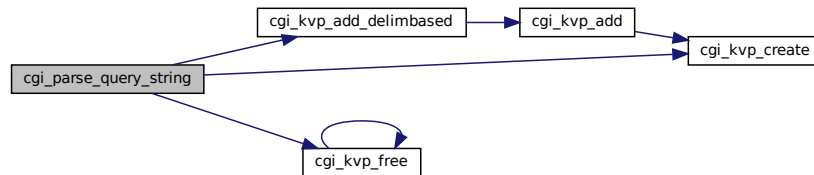
The values can be accessed via [cgi_kvp_get\(\)](#).

The returned pointer must be freed using [cgi_kvp_free\(\)](#).

Definition at line 35 of file cgi.c.

References `cgi_kvp_add_delimbased()`, `cgi_kvp_create()`, and `cgi_kvp_free()`.

Here is the call graph for this function:



5.3.2.8 cgi_testenv()

```

int cgi_testenv (
    char * errmsg,
    char method,
    const char * post_content_type )
  
```

Parameters

<code>in</code>	<code>method</code>	'p': POST 'g': GET
-----------------	---------------------	--------------------

Definition at line 280 of file cgi.c.

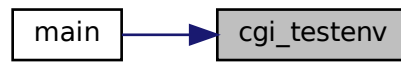
References `CGI_MAXINPUTSIZE`, and `isprefix()`.

Referenced by `main()`.

Here is the call graph for this function:



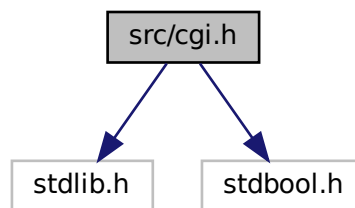
Here is the caller graph for this function:



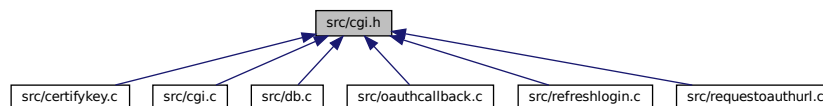
5.4 src/cgi.h File Reference

Header file for [cgi.c](#).

```
#include <stdlib.h>
#include <stdbool.h>
Include dependency graph for cgi.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [cgi_kvps](#)
key-value-pairs struct

Macros

- #define `CGI_MAXINPUTSIZE` 65536
the maximum value for CONTENT_LENGTH

Functions

- struct `cgi_kvp_s` * `cgi_parse_query_string` (const char *qrstr)
Parses a QUERY_STRING.
- struct `cgi_kvp_s` * `cgi_parse_http_cookie` (const char *cokstr)
Parses a HTTP_COOKIE string.
- struct `cgi_kvp_s` * `cgi_kvp_create` (void)
Create a key-value-pairs list.
- void `cgi_kvp_free` (struct `cgi_kvp_s` *kvp)
Frees a key-value-pairs list.
- bool `cgi_kvp_add` (struct `cgi_kvp_s` *kvp, const char *key, const char *value)
Adds a key-value-pair to a key-value-pairs list.
- bool `cgi_kvp_add_delimbased` (struct `cgi_kvp_s` *kvp, char *keyvalue, const char delim)
Adds a key-value-pair to a key-value-pairs list but the key-value-pair is represented as a single string seperated by a delimiter.
- char * `cgi_kvp_get` (struct `cgi_kvp_s` *kvp, const char *key)
Gets a value associated with a given key from a key-value-pairs list.
- int `cgi_testenv` (char *errmsg, char method, const char *post_content_type)
- void `cgi_printstatuscodeasstr` (int statuscode)
- void `cgi_printerror` (char *errmsg, int statuscode)

5.4.1 Detailed Description

Header file for `cgi.c`.

5.4.2 Function Documentation

5.4.2.1 `cgi_kvp_add()`

```
bool cgi_kvp_add (  
    struct cgi_kvp_s * kvp,  
    const char * key,  
    const char * value )
```

Adds a key-value-pair to a key-value-pairs list.

Note

Neither key or value can be NULL.

Duplicate keys can exist and `cgi_kvp_get()` will only retrieve the first one. Thus you cannot override a value.

Parameters

<i>in, out</i>	<i>kvp</i>	the key-value-pairs list
<i>in</i>	<i>key</i>	the key
<i>in</i>	<i>value</i>	the value

Returns

whether the pair was successfully added to the list

Return values

<i>true</i>	success
<i>false</i>	an error occured and errno might be set

Definition at line 179 of file cgi.c.

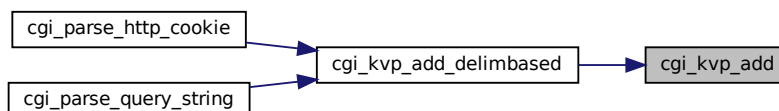
References `cgi_kvp_create()`, `cgi_kvp_s::key`, `cgi_kvp_s::next`, and `cgi_kvp_s::value`.

Referenced by `cgi_kvp_add_delimbased()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.2 cgi_kvp_add_delimbased()

```
bool cgi_kvp_add_delimbased (
    struct cgi_kvp_s * kvp,
    char * keyvalue,
    const char delim )
```

Adds a key-value-pair to a key-value-pairs list but the key-value-pair is represented as a single string seperated by a delimiter.

If the delimiter is not or more than once in keyvalue, then 0 is returned.

This function is destructive and will replace the delimiter in keyvalue with '\0'.

The delimiter cannot be '\0'.

See also

[cgi_kvp_add\(\)](#)

Parameters

in, out	<i>kvp</i>	the key-value-pairs list
in	<i>keyvalue</i>	the key and value in the same string seperated by the delimiter
in	<i>delim</i>	the delimiter

Returns

whether the pair was successfully added to the list

Return values

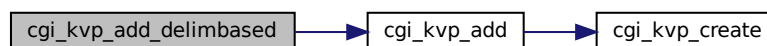
<i>true</i>	success
<i>false</i>	an error ocured and errno might be set or the delimiter was not exactly once in the string

Definition at line 235 of file cgi.c.

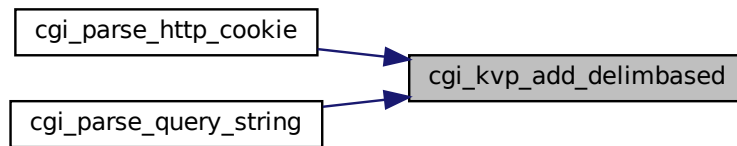
References [cgi_kvp_add\(\)](#).

Referenced by [cgi_parse_http_cookie\(\)](#), and [cgi_parse_query_string\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.3 `cgi_kvp_create()`

```
struct cgi_kvp_s* cgi_kvp_create (
    void )
```

Create a key-value-pairs list.

Returns

a key-value-pairs struct

Return values

<code>NULL</code>	an error occurred and <code>errno</code> is set
-------------------	---

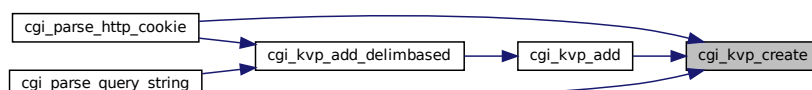
The returned pointer must be freed using `cgi_kvp_free()`.

Definition at line 131 of file `cgi.c`.

References `cgi_kvp_s::key`, `cgi_kvp_s::next`, and `cgi_kvp_s::value`.

Referenced by `cgi_kvp_add()`, `cgi_parse_http_cookie()`, and `cgi_parse_query_string()`.

Here is the caller graph for this function:



5.4.2.4 cgi_kvp_free()

```
void cgi_kvp_free (
    struct cgi_kvp_s * kvp )
```

Frees a key-value-pairs list.

Parameters

in, out	kvp	the key-value-pairs list
---------	-----	--------------------------

See also

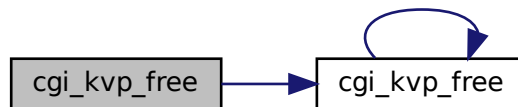
[cgi_kvp_create\(\)](#)

Definition at line 152 of file cgi.c.

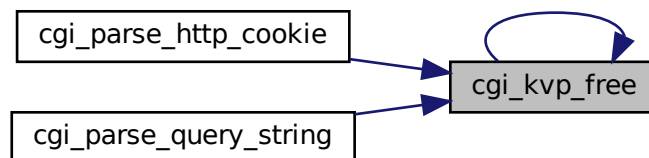
References [cgi_kvp_free\(\)](#), [cgi_kvp_s::key](#), [cgi_kvp_s::next](#), and [cgi_kvp_s::value](#).

Referenced by [cgi_kvp_free\(\)](#), [cgi_parse_http_cookie\(\)](#), and [cgi_parse_query_string\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.5 `cgi_kvp_get()`

```
char* cgi_kvp_get (
    struct cgi_kvp_s * kvp,
    const char * key )
```

Gets a value associated with a given key from a key-value-pairs list.

Parameters

<code>in, out</code>	<code>kvp</code>	the key-value-pairs list
<code>in</code>	<code>key</code>	the key

Returns

the value associated with the key

Return values

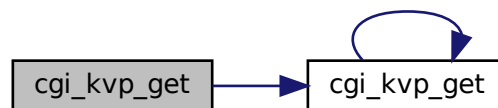
<code>NULL</code>	no value is associated with the key
-------------------	-------------------------------------

Definition at line 262 of file `cgi.c`.

References `cgi_kvp_get()`, `cgi_kvp_s::key`, `cgi_kvp_s::next`, and `cgi_kvp_s::value`.

Referenced by `cgi_kvp_get()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.2.6 `cgi_parse_http_cookie()`

```
struct cgi_kvp_s* cgi_parse_http_cookie (
    const char * cokstr )
```

Parses a HTTP_COOKIE string.

This is not standard conforming!

The HTTP_COOKIE sting is a a list of key-value-pairs seperated by ';' characters. Each key-value-pair is intern seperated by an '=' character. Whitespaces are ignored.

Parameters

<code>in</code>	<code>cokstr</code>	the HTTP_COOKIE string.
-----------------	---------------------	-------------------------

Returns

a key-value-pairs list containing the key-value-pairs

Return values

<code>NULL</code>	an error occurred and <code>errno</code> might be set. maybe the HTTP_COOKIE string is invalid?
-------------------	---

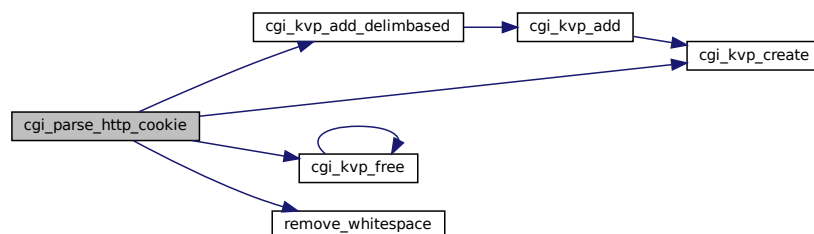
The values can be accessed via [`cgi_kvp_get\(\)`](#).

The returned pointer must be freed using [`cgi_kvp_free\(\)`](#).

Definition at line 86 of file `cgi.c`.

References [`cgi_kvp_add_delimbased\(\)`](#), [`cgi_kvp_create\(\)`](#), [`cgi_kvp_free\(\)`](#), and [`remove_whitespace\(\)`](#).

Here is the call graph for this function:



5.4.2.7 cgi_parse_query_string()

```
struct cgi_kvp_s* cgi_parse_query_string (
    const char * qrystr )
```

Parses a QUERY_STRING.

This is not standard conforming!

The QUERY_STRING is a a list of key-value-pairs separated by '&' characters. Each key-value-pair is intern seperated by an '=' character.

Parameters

in	<i>qrystr</i>	the QUERY_STRING
----	---------------	------------------

Returns

a key-value-pairs list containing the key-value-pairs

Return values

<i>NULL</i>	an error occurred and errno might be set. maybe the QUERY_STRING is invalid?
-------------	--

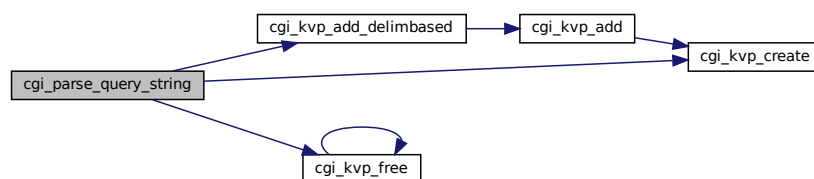
The values can be accessed via [cgi_kvp_get\(\)](#).

The returned pointer must be freed using [cgi_kvp_free\(\)](#).

Definition at line 35 of file cgi.c.

References [cgi_kvp_add_delimbased\(\)](#), [cgi_kvp_create\(\)](#), and [cgi_kvp_free\(\)](#).

Here is the call graph for this function:



5.4.2.8 cgi_testenv()

```
int cgi_testenv (
    char * errmsg,
    char method,
    const char * post_content_type )
```

Parameters

in	method	'p': POST 'g': GET
----	--------	--------------------

Definition at line 280 of file cgi.c.

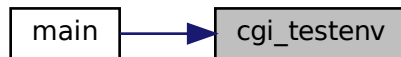
References CGI_MAXINPUTSIZE, and isprefix().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5 src/common.c File Reference

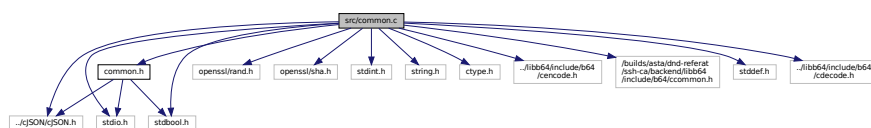
A bunch of useful/common code.

```

#include "common.h"
#include <openssl/rand.h>
#include <openssl/sha.h>
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include <string.h>
#include <ctype.h>
#include "../cJSON/cJSON.h"
#include "../libb64/include/b64/cencode.h"
#include "../libb64/include/b64/cdecode.h"

```

Include dependency graph for common.c:



Functions

- bool `uuid_v4_gen` (char *buffer)
Generate a Version 4 UUID according to RFC-4122.
- bool `is_valid_uuid_v4` (const char *buffer)
Tests whether a given string is a valid UUIDv4.
- bool `is_valid_hex` (char c)
Tests whether a character is a valid hexadecimal character.
- int `hex_to_int` (char c)
Converts a hexadecimal character to a number.
- char `int_to_hex` (int x)
Converts a number to an uppercase hexadecimal character.
- bool `is_percent_encoded` (const char *str)
Tests whether a given string is correctly percent encoded.
- char * `percent_decode` (const char *str)
Decodes a percent encoded string.
- char * `percent_encode` (const char *str)
Percent encodes a string.
- bool `isprefix` (const char *pre, const char *str)
Tests whether str starts with pre.
- char * `filetostr` (const char *path)
Reads a file into a string.
- cJSON * `filetojson` (const char *path)
Reads a JSON file.
- char * `base64_encode` (const char *str, size_t len)
base64 encodes a string
- char * `base64_decode` (const char *str)
decodes a base64 encoded string
- char * `base64url_encode` (const char *str, size_t len, bool skippadding)
base64url encodes a string
- char * `base64url_decode` (char *str)
decodes a base64url encoded string
- bool `hmac_sha256` (unsigned char *hash, const char *message, const char *key)
Calculates the HMAC-SHA256 of a given message with a given key.
- void `remove_whitespace` (char *s)
Removes all whitespace characters from a string.
- bool `writestreamtofile` (FILE *stream, const char *outpath)
Writes a stream to the file located at outpath.
- bool `get_string_from_json` (char *errmsg, const cJSON *json, const char *curjsonkey, const char *stringkey, char **res, bool can_be_null)
Tries to retrieve the value of stringkey from json as a string.
- bool `get_number_from_json` (char *errmsg, const cJSON *json, const char *curjsonkey, const char *numberkey, long *res)
Tries to retrieve the value of numberkey from json as a long.
- bool `get_bool_from_json` (char *errmsg, const cJSON *json, const char *curjsonkey, const char *boolkey, bool *res)
Tries to retrieve the value of boolkey from json as an int.
- char * `str_append` (char *dest, const char *src, size_t *destlen, size_t srclen)
Resizes the dest string and appends the src string to the dest string.

5.5.1 Detailed Description

A bunch of useful/common code.

5.5.2 Function Documentation

5.5.2.1 base64_decode()

```
char* base64_decode (  
    const char * str )
```

decodes a base64 encoded string

Uses libb64 to decode a string.

Parameters

in	str	the encoded input string to be decoded
----	-----	--

Returns

the decoded string

Return values

NULL	an error occurred and errno might be set
------	--

The returned string must be freed using free().

Definition at line 477 of file common.c.

Referenced by base64url_decode().

Here is the caller graph for this function:



5.5.2.2 base64_encode()

```
char* base64_encode (
    const char * str,
    size_t len )
```

base64 encodes a string

Uses libb64 to encode a string.

Parameters

in	<i>str</i>	the input string to be encoded
in	<i>len</i>	the length of the string (if str isn't raw binary this should be strlen(str))

Returns

the encoded string

Return values

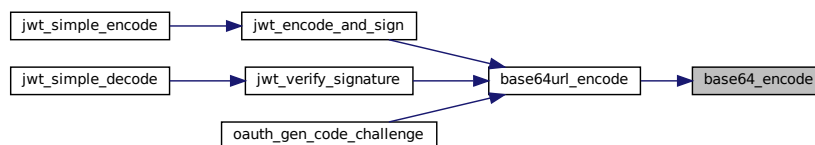
<i>NULL</i>	an error occurred and errno might be set
-------------	--

The returned string must be freed using free().

Definition at line 419 of file common.c.

Referenced by base64url_encode().

Here is the caller graph for this function:



5.5.2.3 base64url_decode()

```
char* base64url_decode (
    char * str )
```

decodes a base64url encoded string

Conforming to [RFC 4648 §5](#).

This function will change `str`! This is because you will probably use this function with the result of a [percent_decode\(\)](#) call.

Note

This function expects the padding to be using '=' characters.

Parameters

<code>in, out</code>	<code>str</code>	the encoded input string to be decoded
----------------------	------------------	--

Returns

the decoded string

Return values

<code>NULL</code>	an error occurred and <code>errno</code> might be set
-------------------	---

The returned string must be freed using `free()`.

Definition at line 574 of file `common.c`.

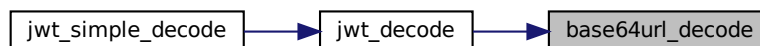
References `base64_decode()`.

Referenced by `jwt_decode()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**5.5.2.4 base64url_encode()**

```

char* base64url_encode (
    const char * str,
    size_t len,
    bool skippadding )
  
```

`base64url` encodes a string

Conforming to [RFC 4648 §5](#).

Note

You might wish to use [percent_encode\(\)](#) on the result of this function before passing it over into a url, as padding is still using '=' characters. Or you just disable padding with `skippadding`.

Parameters

in	<i>str</i>	the input string to be encoded
in	<i>len</i>	the length of the string (if str isn't raw binary this should be <code>strlen(str)</code>)
in	<i>skippadding</i>	whether the padding at the end should be skipped

Returns

the encoded string

Return values

<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

The returned string must be freed using `free()`.

Definition at line 538 of file `common.c`.

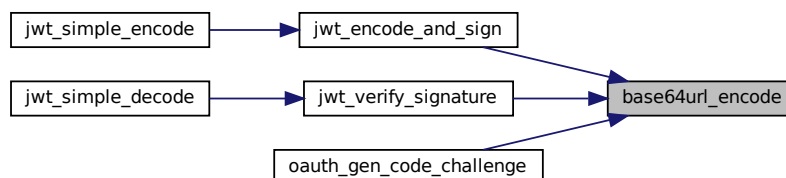
References `base64_encode()`.

Referenced by `jwt_encode_and_sign()`, `jwt_verify_signature()`, and `oauth_gen_code_challenge()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.2.5 filetojson()

```
cJSON* filetojson (
    const char * path )
```

Reads a JSON file.

Reads the file at path and parses it into a cJSON object.

Returns

The files contents as a cJSON object.

Return values

<i>NULL</i>	an error occurred and errno might be set.
-------------	---

The returned cJSON object must be freed using cJSON_Delete().

Definition at line 395 of file common.c.

References filetostr().

Referenced by config_parse().

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.2.6 filetostr()

```
char* filetostr (
    const char * path )
```

Reads a file into a string.

Reads the file at path and converts it into a null-terminated string.

Note

This function is not suitable for binary files that may contain '\0' characters.

Returns

The files contents as a string.

Return values

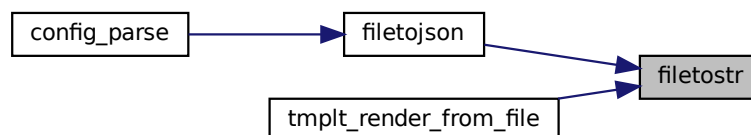
<i>NULL</i>	an error occurred and errno might be set.
-------------	---

The returned string must be freed using free().

Definition at line 358 of file common.c.

Referenced by filetojson(), and tmpl_t_render_from_file().

Here is the caller graph for this function:



5.5.2.7 get_bool_from_json()

```
bool get_bool_from_json (
    char * errmsg,
    const cJSON * json,
    const char * curjsonkey,
    const char * boolkey,
    bool * res )
```

Tries to retrieve the value of boolkey from json as an int.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>json</i>	the json object that is supposed to contain boolkey
in	<i>curjsonkey</i>	the name of json (used for error messages)
in	<i>boolkey</i>	the name of the key
out	<i>res</i>	where the retrieved value should be stored

Returns

whether the function was successful.

Return values

<i>true</i>	success. a pointer to the value has been stored in res
<i>false</i>	failed retrieving the bool and an error message has been written to errmsg

Definition at line 839 of file common.c.

5.5.2.8 get_number_from_json()

```
bool get_number_from_json (
    char * errmsg,
    const cJSON * json,
    const char * curjsonkey,
    const char * numberkey,
    long * res )
```

Tries to retrieve the value of numberkey from json as a long.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>json</i>	the json object that is supposed to contain numberkey
in	<i>curjsonkey</i>	the name of json (used for error messages)
in	<i>numberkey</i>	the name of the key
out	<i>res</i>	where the retrieved value should be stored

Returns

whether the function was successful.

Return values

<i>true</i>	success. a pointer to the value has been stored in res
<i>false</i>	failed retrieving the number and an error message has been written to errmsg

Definition at line 802 of file common.c.

5.5.2.9 get_string_from_json()

```
bool get_string_from_json (
    char * errmsg,
    const cJSON * json,
    const char * curjsonkey,
    const char * stringkey,
    char ** res,
    bool can_be_null )
```

Tries to retrieve the value of stringkey from json as a string.

Note: res will only contain the pointer to the string inside the json object and thus will be freed by calling cJSON_Delete on json.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>json</i>	the json object that is supposed to contain stringkey
in	<i>curjsonkey</i>	the name of json (used for error messages)
in	<i>stringkey</i>	the name of the key
out	<i>res</i>	pointer to the pointer where the pointer of the retrieved value should be stored
in	<i>can_be_null</i>	whether the value is allowed to be null

Returns

whether the function was successful.

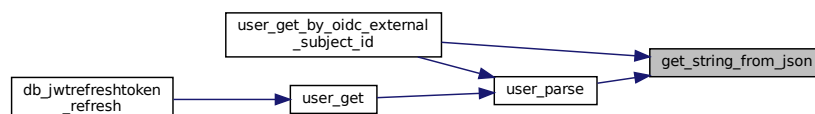
Return values

<i>true</i>	success. a pointer to the value has been stored in res
<i>false</i>	failed retrieving the string and an error message has been written to errmsg

Definition at line 758 of file common.c.

Referenced by user_get_by_oidc_external_subject_id(), and user_parse().

Here is the caller graph for this function:



5.5.2.10 hex_to_int()

```
int hex_to_int (
    char c )
```

Converts a hexadecimal character to a number.

Basically '0'-'9' are mapped to 0-9 and 'a'-'f' or 'A'-'F' are mapped to 10-15.

A non-hexadecimal character is just treated as '0'.

Parameters

in	<i>c</i>	the character
----	----------	---------------

Returns

c as a number

Definition at line 150 of file common.c.

5.5.2.11 hmac_sha256()

```
bool hmac_sha256 (
    unsigned char * hash,
    const char * message,
    const char * key )
```

Calculates the HMAC-SHA256 of a given message with a given key.

See also

<https://en.wikipedia.org/wiki/HMAC>

Parameters

out	<i>hash</i>	buffer for the resulting hash; Must be at least SHA256_DIGEST_LENGTH bytes long.
in	<i>message</i>	the message
in	<i>key</i>	the key

Returns

whether the function was successful.

Return values

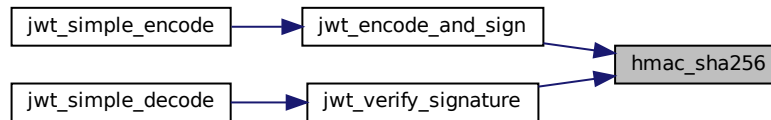
<i>true</i>	success
<i>false</i>	an error occurred and errno might be set

Definition at line 609 of file common.c.

References `cgi_kvps::key`.

Referenced by `jwt_encode_and_sign()`, and `jwt_verify_signature()`.

Here is the caller graph for this function:



5.5.2.12 int_to_hex()

```
char int_to_hex (  
    int x )
```

Converts a number to an uppercase hexadecimal character.

Basically 0-9 are mapped to '0'-'9' and 10-15 are mapped to 'A'-'F'.

A number larger than 15 or smaller than 0 is just treated as 0.

Parameters

in	x	the number
----	---	------------

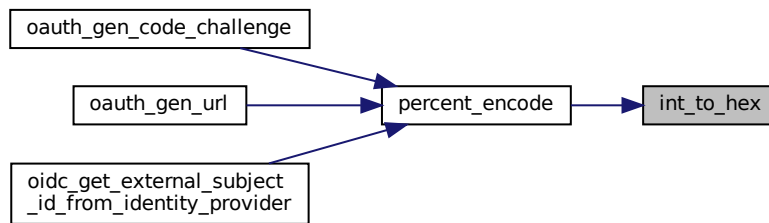
Returns

x as an uppercase hexadecimal character

Definition at line 173 of file common.c.

Referenced by `percent_encode()`.

Here is the caller graph for this function:



5.5.2.13 `is_percent_encoded()`

```
bool is_percent_encoded (
    const char * str )
```

Tests whether a given string is correctly percent encoded.

Conforming to [RFC 3986 §2.1](#).

Parameters

<code>in</code>	<code>str</code>	the string to be tested
-----------------	------------------	-------------------------

Returns

whether `str` is correctly percent encoded.

Definition at line 190 of file `common.c`.

References `is_valid_hex()`.

Here is the call graph for this function:



5.5.2.14 is_valid_hex()

```
bool is_valid_hex (
    char c )
```

Tests whether a character is a valid hexadecimal character.

Valid hexadecimal characters are: 0-9 a-f A-F

Parameters

in	c	the character to be tested
----	---	----------------------------

Returns

whether c is a valid hex character.

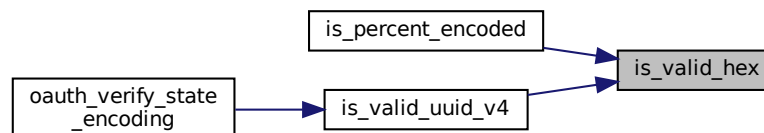
Return values

<i>false</i>	not a valid UUIDv4
<i>true</i>	a valid UUIDv4

Definition at line 132 of file common.c.

Referenced by `is_percent_encoded()`, and `is_valid_uuid_v4()`.

Here is the caller graph for this function:



5.5.2.15 is_valid_uuid_v4()

```
bool is_valid_uuid_v4 (
    const char * buffer )
```

Tests whether a given string is a valid UUIDv4.

A UUIDv4 looks like this: 034d9d19-0182-4fd8-aa18-21fd18bf956

See also

[uuid_v4_gen\(\)](#)

Parameters

in	<i>buffer</i>	the string to be tested
----	---------------	-------------------------

Returns

whether *buffer* is a valid UUIDv4.

Return values

<i>false</i>	not a valid UUIDv4
<i>true</i>	a valid UUIDv4

Definition at line 82 of file common.c.

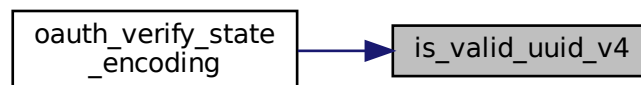
References `is_valid_hex()`, and `UUIDLEN`.

Referenced by `oauth_verify_state_encoding()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**5.5.2.16 isprefix()**

```

bool isprefix (
    const char * pre,
    const char * str )
  
```

Tests whether *str* starts with *pre*.

Parameters

<code>in</code>	<code>pre</code>	the expected prefix string
<code>in</code>	<code>str</code>	the string to be tested

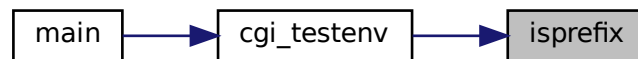
Returns

whether `str` starts with `pre`.

Definition at line 341 of file `common.c`.

Referenced by `cgi_testenv()`.

Here is the caller graph for this function:

**5.5.2.17 percent_decode()**

```
char* percent_decode (  
    const char * str )
```

Decodes a percent encoded string.

This function expects `str` to be conforming to [RFC 3986 §2.1](#). You can test if `str` is valid using [is_percent_encoded\(\)](#).

Note

The sequence `"%00"` may cause undefined behaviour as it will just null-terminate the string at that point.

Parameters

<code>in</code>	<code>str</code>	the percent encoded string to be decoded
-----------------	------------------	--

Returns

the decoded string

Return values

<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

The returned string must be freed using `free()`.

Definition at line 251 of file `common.c`.

5.5.2.18 percent_encode()

```
char* percent_encode (  
    const char * str )
```

Percent encodes a string.

Conforming to [RFC 3986 §2.1](#).

Note

This function cannot handle `'\0'` characters as those are used for indicating the end of a string in C.

Parameters

<i>in</i>	<i>str</i>	the string to be encoded
-----------	------------	--------------------------

Returns

the encoded string

Return values

<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

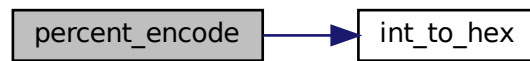
The returned string must be freed using `free()`.

Definition at line 296 of file `common.c`.

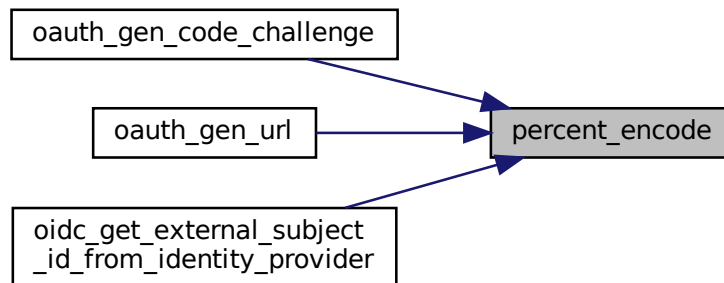
References `int_to_hex()`.

Referenced by `oauth_gen_code_challenge()`, `oauth_gen_url()`, and `oidc_get_external_subject_id_from_identity_↔ provider()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.2.19 remove_whitespace()

```
void remove_whitespace (  
    char * s )
```

Removes all whitespace characters from a string.

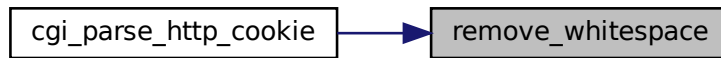
Parameters

in, out	s	the string
---------	---	------------

Definition at line 680 of file common.c.

Referenced by `cgi_parse_http_cookie()`.

Here is the caller graph for this function:



5.5.2.20 str_append()

```

char* str_append (
    char * dest,
    const char * src,
    size_t * destlen,
    size_t srclen )
  
```

Resizes the dest string and appends the src string to the dest string.

Uses realloc() to resize dest to fit dest, src and the terminating null byte, then it appends srclen characters of the src string to the dest string, overwriting the terminating null byte at the end of dest, and then adds a terminating null byte.

Note

dest and src must not be NULL.

Parameters

in, out	<i>dest</i>	the destination string
in	<i>src</i>	the source string
in, out	<i>destlen</i>	pointer where the current length of dest is stored. This function also updates this value. Excluding the terminating null byte. Ignored if NULL.
in	<i>srclen</i>	the length of src. This function will use strlen(src), if srclen == 0.

Returns

the new location of dest.

Return values

NULL	an error occurred and errno might be set.
------	---

Note

dest is freed by this function if an error occurs.

Definition at line 884 of file common.c.

Referenced by `tmpl_render()`.

Here is the caller graph for this function:

**5.5.2.21 uuid_v4_gen()**

```
bool uuid_v4_gen (
    char * buffer )
```

Generate a Version 4 UUID according to RFC-4122.

Uses the openssl `RAND_bytes` function to generate a Version 4 UUID.

Parameters

<i>in, out</i>	<i>buffer</i>	A buffer that is at least (UUIDLEN + 1) bytes long.
----------------	---------------	---

Return values

<i>true</i>	success
<i>false</i>	failure

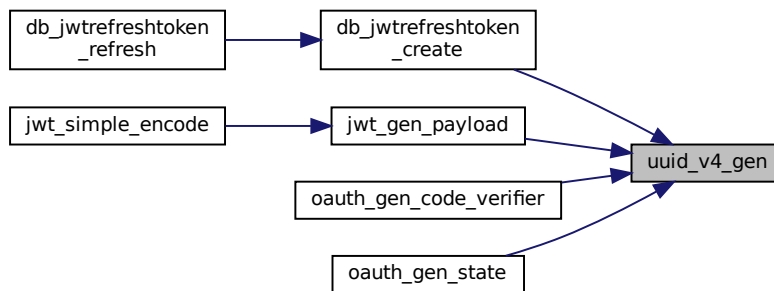
Note

Code from <https://gist.github.com/kvelakur/9069c9896577c3040030>

Definition at line 35 of file common.c.

Referenced by `db_jwtrefresh_token_create()`, `jwt_gen_payload()`, `oauth_gen_code_verifier()`, and `oauth_gen_state()`.

Here is the caller graph for this function:



5.5.2.22 writestreamtofile()

```

bool writestreamtofile (
    FILE * stream,
    const char * outpath )
  
```

Writes a stream to the file located at outpath.

The stream is read until EOF is reached.

Parameters

in	<i>stream</i>	the stream to be read from
in	<i>outpath</i>	path to the file were the output should be written to

Returns

whether the function was successful

Return values

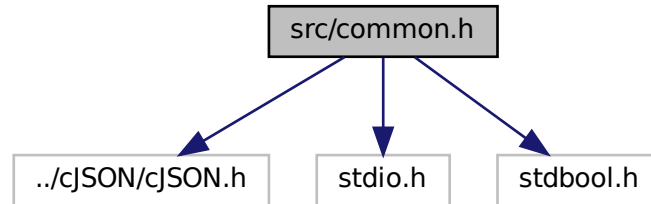
<i>true</i>	success
<i>false</i>	An error occurred and errno might be set

Definition at line 711 of file common.c.

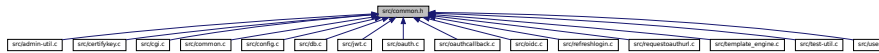
5.6 src/common.h File Reference

Header file for [common.c](#).

```
#include "../cJSON/cJSON.h"
#include <stdio.h>
#include <stdbool.h>
Include dependency graph for common.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- #define **COLOR_BLACK** "\033[0;30m"
ANSI Escape Sequence for the color black.
- #define **COLOR_RED** "\033[0;31m"
ANSI Escape Sequence for the color red.
- #define **COLOR_GREEN** "\033[0;32m"
ANSI Escape Sequence for the color green.
- #define **COLOR_YELLOW** "\033[0;33m"
ANSI Escape Sequence for the color yellow.
- #define **COLOR_BLUE** "\033[0;34m"
ANSI Escape Sequence for the color blue.
- #define **COLOR_PURPLE** "\033[0;35m"
ANSI Escape Sequence for the color purple.
- #define **COLOR_CYAN** "\033[0;36m"
ANSI Escape Sequence for the color cyan.
- #define **COLOR_WHITE** "\033[0;37m"
ANSI Escape Sequence for the color white.
- #define **COLOR_RESET** "\033[0m"
ANSI Escape Sequence for resetting the color.
- #define **CONFIGFILEPATH** "/opt/ssh-ca/backend/config.json"
path to the config file
- #define **ERRMSGMAXSIZE** 1024
size of errmsg buffer
- #define **UUIDLEN** 36
the length of a UUID
- #define **UNUSED(x)** (void)(x)
macro for marking a variable as unused

Functions

- bool `uuid_v4_gen` (char *buffer)
Generate a Version 4 UUID according to RFC-4122.
- bool `is_valid_uuid_v4` (const char *buffer)
Tests whether a given string is a valid UUIDv4.
- bool `is_valid_hex` (char c)
Tests whether a character is a valid hexadecimal character.
- int `hex_to_int` (char c)
Converts a hexadecimal character to a number.
- char `int_to_hex` (int x)
Converts a number to an uppercase hexadecimal character.
- bool `is_percent_encoded` (const char *str)
Tests whether a given string is correctly percent encoded.
- char * `percent_decode` (const char *str)
Decodes a percent encoded string.
- char * `percent_encode` (const char *str)
Percent encodes a string.
- bool `isprefix` (const char *pre, const char *str)
Tests whether str starts with pre.
- char * `filetostr` (const char *path)
Reads a file into a string.
- cJSON * `filetojson` (const char *path)
Reads a JSON file.
- char * `base64_encode` (const char *str, size_t len)
base64 encodes a string
- char * `base64_decode` (const char *str)
decodes a base64 encoded string
- char * `base64url_encode` (const char *str, size_t len, bool skippadding)
base64url encodes a string
- char * `base64url_decode` (char *str)
decodes a base64url encoded string
- bool `hmac_sha256` (unsigned char *hash, const char *message, const char *key)
Calculates the HMAC-SHA256 of a given message with a given key.
- void `remove_whitespace` (char *s)
Removes all whitespace characters from a string.
- bool `writestreamtofile` (FILE *stream, const char *outpath)
Writes a stream to the file located at outpath.
- bool `get_string_from_json` (char *errmsg, const cJSON *json, const char *curjsonkey, const char *stringkey, char **res, bool can_be_null)
Tries to retrieve the value of stringkey from json as a string.
- bool `get_number_from_json` (char *errmsg, const cJSON *json, const char *curjsonkey, const char *numberkey, long *res)
Tries to retrieve the value of numberkey from json as a long.
- bool `get_bool_from_json` (char *errmsg, const cJSON *json, const char *curjsonkey, const char *boolkey, bool *res)
Tries to retrieve the value of boolkey from json as an int.
- char * `str_append` (char *dest, const char *src, size_t *destlen, size_t srclen)
Resizes the dest string and appends the src string to the dest string.

5.6.1 Detailed Description

Header file for [common.c](#).

5.6.2 Function Documentation

5.6.2.1 base64_decode()

```
char* base64_decode (  
    const char * str )
```

decodes a base64 encoded string

Uses libb64 to decode a string.

Parameters

in	str	the encoded input string to be decoded
----	-----	--

Returns

the decoded string

Return values

NULL	an error occurred and errno might be set
------	--

The returned string must be freed using `free()`.

Definition at line 477 of file `common.c`.

Referenced by `base64url_decode()`.

Here is the caller graph for this function:



5.6.2.2 base64_encode()

```
char* base64_encode (
    const char * str,
    size_t len )
```

base64 encodes a string

Uses libb64 to encode a string.

Parameters

in	<i>str</i>	the input string to be encoded
in	<i>len</i>	the length of the string (if str isn't raw binary this should be strlen(str))

Returns

the encoded string

Return values

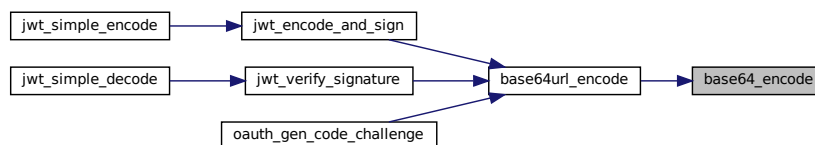
<i>NULL</i>	an error occurred and errno might be set
-------------	--

The returned string must be freed using free().

Definition at line 419 of file common.c.

Referenced by base64url_encode().

Here is the caller graph for this function:



5.6.2.3 base64url_decode()

```
char* base64url_decode (
    char * str )
```

decodes a base64url encoded string

Conforming to [RFC 4648 §5](#).

This function will change str! This is because you will probably use this function with the result of a [percent_decode\(\)](#) call.

Note

This function expects the padding to be using '=' characters.

Parameters

<code>in, out</code>	<code>str</code>	the encoded input string to be decoded
----------------------	------------------	--

Returns

the decoded string

Return values

<code>NULL</code>	an error occurred and <code>errno</code> might be set
-------------------	---

The returned string must be freed using `free()`.

Definition at line 574 of file `common.c`.

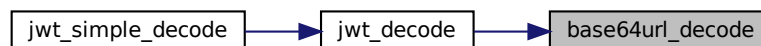
References `base64_decode()`.

Referenced by `jwt_decode()`.

Here is the call graph for this function:



Here is the caller graph for this function:

**5.6.2.4 base64url_encode()**

```
char* base64url_encode (  
    const char * str,  
    size_t len,  
    bool skippadding )
```

`base64url` encodes a string

Conforming to [RFC 4648 §5](#).

Note

You might wish to use [percent_encode\(\)](#) on the result of this function before passing it over into a url, as padding is still using '=' characters. Or you just disable padding with `skippadding`.

Parameters

in	<i>str</i>	the input string to be encoded
in	<i>len</i>	the length of the string (if str isn't raw binary this should be <code>strlen(str)</code>)
in	<i>skippadding</i>	whether the padding at the end should be skipped

Returns

the encoded string

Return values

<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

The returned string must be freed using `free()`.

Definition at line 538 of file `common.c`.

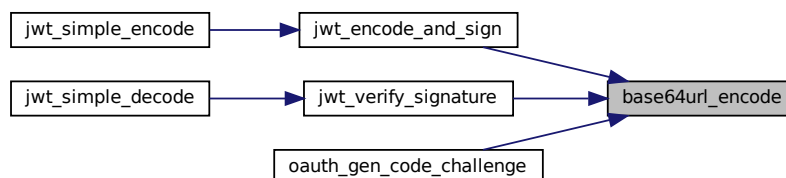
References `base64_encode()`.

Referenced by `jwt_encode_and_sign()`, `jwt_verify_signature()`, and `oauth_gen_code_challenge()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.5 filetojson()

```
cJSON* filetojson (
    const char * path )
```

Reads a JSON file.

Reads the file at path and parses it into a cJSON object.

Returns

The files contents as a cJSON object.

Return values

<i>NULL</i>	an error occurred and errno might be set.
-------------	---

The returned cJSON object must be freed using cJSON_Delete().

Definition at line 395 of file common.c.

References filetostr().

Referenced by config_parse().

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.6 filetostr()

```
char* filetostr (
    const char * path )
```

Reads a file into a string.

Reads the file at path and converts it into a null-terminated string.

Note

This function is not suitable for binary files that may contain '\0' characters.

Returns

The files contents as a string.

Return values

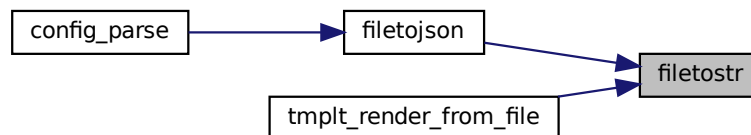
<i>NULL</i>	an error occurred and errno might be set.
-------------	---

The returned string must be freed using free().

Definition at line 358 of file common.c.

Referenced by filetojson(), and tmplt_render_from_file().

Here is the caller graph for this function:



5.6.2.7 get_bool_from_json()

```
bool get_bool_from_json (
    char * errmsg,
    const cJSON * json,
    const char * curjsonkey,
    const char * boolkey,
    bool * res )
```

Tries to retrieve the value of boolkey from json as an int.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>json</i>	the json object that is supposed to contain boolkey
in	<i>curjsonkey</i>	the name of json (used for error messages)
in	<i>boolkey</i>	the name of the key
out	<i>res</i>	where the retrieved value should be stored

Returns

whether the function was successful.

Return values

<i>true</i>	success. a pointer to the value has been stored in res
<i>false</i>	failed retrieving the bool and an error message has been written to errmsg

Definition at line 839 of file common.c.

5.6.2.8 get_number_from_json()

```
bool get_number_from_json (
    char * errmsg,
    const cJSON * json,
    const char * curjsonkey,
    const char * numberkey,
    long * res )
```

Tries to retrieve the value of numberkey from json as a long.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>json</i>	the json object that is supposed to contain numberkey
in	<i>curjsonkey</i>	the name of json (used for error messages)
in	<i>numberkey</i>	the name of the key
out	<i>res</i>	where the retrieved value should be stored

Returns

whether the function was successful.

Return values

<i>true</i>	success. a pointer to the value has been stored in res
<i>false</i>	failed retrieving the number and an error message has been written to errmsg

Definition at line 802 of file common.c.

5.6.2.9 get_string_from_json()

```
bool get_string_from_json (
    char * errmsg,
    const cJSON * json,
    const char * curjsonkey,
    const char * stringkey,
    char ** res,
    bool can_be_null )
```

Tries to retrieve the value of stringkey from json as a string.

Note: res will only contain the pointer to the string inside the json object and thus will be freed by calling cJSON_Delete on json.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>json</i>	the json object that is supposed to contain stringkey
in	<i>curjsonkey</i>	the name of json (used for error messages)
in	<i>stringkey</i>	the name of the key
out	<i>res</i>	pointer to the pointer where the pointer of the retrieved value should be stored
in	<i>can_be_null</i>	whether the value is allowed to be null

Returns

whether the function was successful.

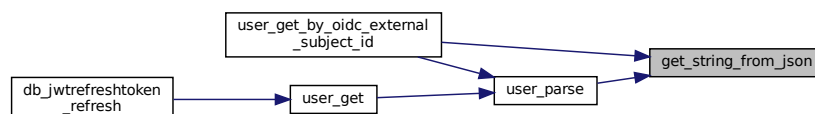
Return values

<i>true</i>	success. a pointer to the value has been stored in res
<i>false</i>	failed retrieving the string and an error message has been written to errmsg

Definition at line 758 of file common.c.

Referenced by user_get_by_oidc_external_subject_id(), and user_parse().

Here is the caller graph for this function:



5.6.2.10 hex_to_int()

```
int hex_to_int (
    char c )
```

Converts a hexadecimal character to a number.

Basically '0'-'9' are mapped to 0-9 and 'a'-'f' or 'A'-'F' are mapped to 10-15.

A non-hexadecimal character is just treated as '0'.

Parameters

in	<i>c</i>	the character
----	----------	---------------

Returns

c as a number

Definition at line 150 of file common.c.

5.6.2.11 hmac_sha256()

```
bool hmac_sha256 (
    unsigned char * hash,
    const char * message,
    const char * key )
```

Calculates the HMAC-SHA256 of a given message with a given key.

See also

<https://en.wikipedia.org/wiki/HMAC>

Parameters

out	<i>hash</i>	buffer for the resulting hash; Must be at least SHA256_DIGEST_LENGTH bytes long.
in	<i>message</i>	the message
in	<i>key</i>	the key

Returns

whether the function was successful.

Return values

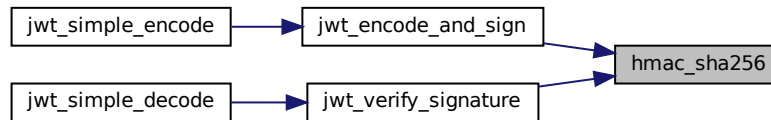
<i>true</i>	success
<i>false</i>	an error occurred and errno might be set

Definition at line 609 of file common.c.

References `cgi_kvps::key`.

Referenced by `jwt_encode_and_sign()`, and `jwt_verify_signature()`.

Here is the caller graph for this function:



5.6.2.12 int_to_hex()

```
char int_to_hex (  
    int x )
```

Converts a number to an uppercase hexadecimal character.

Basically 0-9 are mapped to '0'-'9' and 10-15 are mapped to 'A'-'F'.

A number larger than 15 or smaller than 0 is just treated as 0.

Parameters

in	x	the number
----	---	------------

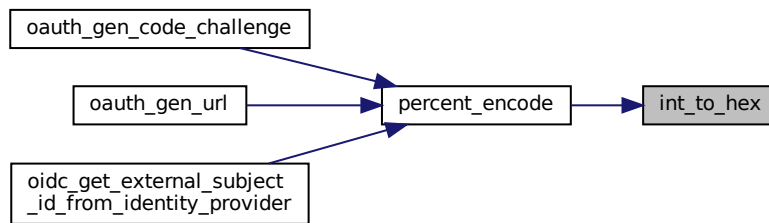
Returns

x as an uppercase hexadecimal character

Definition at line 173 of file common.c.

Referenced by `percent_encode()`.

Here is the caller graph for this function:



5.6.2.13 `is_percent_encoded()`

```
bool is_percent_encoded (
    const char * str )
```

Tests whether a given string is correctly percent encoded.

Conforming to [RFC 3986 §2.1](#).

Parameters

in	<i>str</i>	the string to be tested
----	------------	-------------------------

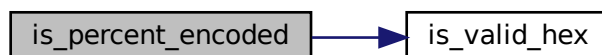
Returns

whether `str` is correctly percent encoded.

Definition at line 190 of file `common.c`.

References `is_valid_hex()`.

Here is the call graph for this function:



5.6.2.14 is_valid_hex()

```
bool is_valid_hex (
    char c )
```

Tests whether a character is a valid hexadecimal character.

Valid hexadecimal characters are: 0-9 a-f A-F

Parameters

in	c	the character to be tested
----	---	----------------------------

Returns

whether c is a valid hex character.

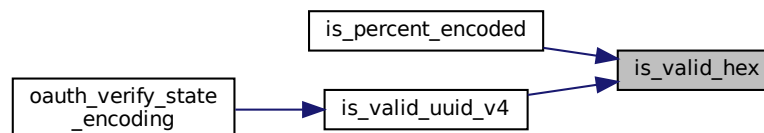
Return values

<i>false</i>	not a valid UUIDv4
<i>true</i>	a valid UUIDv4

Definition at line 132 of file common.c.

Referenced by `is_percent_encoded()`, and `is_valid_uuid_v4()`.

Here is the caller graph for this function:



5.6.2.15 is_valid_uuid_v4()

```
bool is_valid_uuid_v4 (
    const char * buffer )
```

Tests whether a given string is a valid UUIDv4.

A UUIDv4 looks like this: 034d9d19-0182-4fd8-aa18-21fd18bf956

See also

[uuid_v4_gen\(\)](#)

Parameters

in	<i>buffer</i>	the string to be tested
----	---------------	-------------------------

Returns

whether *buffer* is a valid UUIDv4.

Return values

<i>false</i>	not a valid UUIDv4
<i>true</i>	a valid UUIDv4

Definition at line 82 of file common.c.

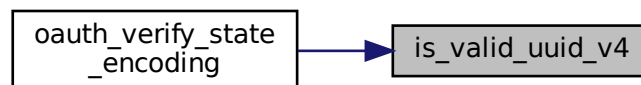
References `is_valid_hex()`, and `UUIDLEN`.

Referenced by `oauth_verify_state_encoding()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.16 isprefix()

```

bool isprefix (
    const char * pre,
    const char * str )
  
```

Tests whether *str* starts with *pre*.

Parameters

in	<i>pre</i>	the expected prefix string
in	<i>str</i>	the string to be tested

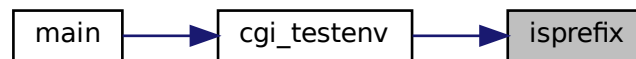
Returns

whether *str* starts with *pre*.

Definition at line 341 of file `common.c`.

Referenced by `cgi_testenv()`.

Here is the caller graph for this function:

**5.6.2.17 percent_decode()**

```
char* percent_decode (
    const char * str )
```

Decodes a percent encoded string.

This function expects *str* to be conforming to [RFC 3986 §2.1](#). You can test if *str* is valid using [is_percent_encoded\(\)](#).

Note

The sequence "%00" may cause undefined behaviour as it will just null-terminate the string at that point.

Parameters

in	<i>str</i>	the percent encoded string to be decoded
----	------------	--

Returns

the decoded string

Return values

<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

The returned string must be freed using `free()`.

Definition at line 251 of file `common.c`.

5.6.2.18 percent_encode()

```
char* percent_encode (  
    const char * str )
```

Percent encodes a string.

Conforming to [RFC 3986 §2.1](#).

Note

This function cannot handle `'\0'` characters as those are used for indicating the end of a string in C.

Parameters

<i>in</i>	<i>str</i>	the string to be encoded
-----------	------------	--------------------------

Returns

the encoded string

Return values

<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

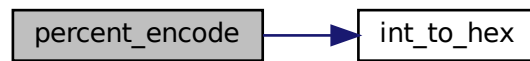
The returned string must be freed using `free()`.

Definition at line 296 of file `common.c`.

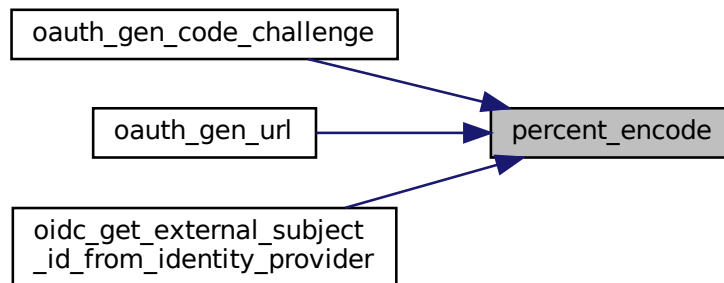
References `int_to_hex()`.

Referenced by `oauth_gen_code_challenge()`, `oauth_gen_url()`, and `oidc_get_external_subject_id_from_identity_provider()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.2.19 remove_whitespace()

```
void remove_whitespace (  
    char * s )
```

Removes all whitespace characters from a string.

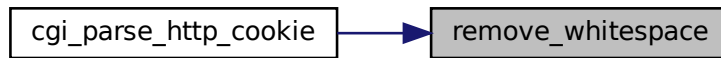
Parameters

in, out	s	the string
---------	---	------------

Definition at line 680 of file common.c.

Referenced by `cgi_parse_http_cookie()`.

Here is the caller graph for this function:



5.6.2.20 str_append()

```

char* str_append (
    char * dest,
    const char * src,
    size_t * destlen,
    size_t srclen )
  
```

Resizes the dest string and appends the src string to the dest string.

Uses realloc() to resize dest to fit dest, src and the terminating null byte, then it appends srclen characters of the src string to the dest string, overwriting the terminating null byte at the end of dest, and then adds a terminating null byte.

Note

dest and src must not be NULL.

Parameters

in, out	<i>dest</i>	the destination string
in	<i>src</i>	the source string
in, out	<i>destlen</i>	pointer where the current length of dest is stored. This function also updates this value. Excluding the terminating null byte. Ignored if NULL.
in	<i>srclen</i>	the length of src. This function will use strlen(src), if srclen == 0.

Returns

the new location of dest.

Return values

NULL	an error occurred and errno might be set.
------	---

Note

dest is freed by this function if an error occurs.

Definition at line 884 of file common.c.

Referenced by `tmpl_render()`.

Here is the caller graph for this function:

**5.6.2.21 uuid_v4_gen()**

```
bool uuid_v4_gen (
    char * buffer )
```

Generate a Version 4 UUID according to RFC-4122.

Uses the openssl `RAND_bytes` function to generate a Version 4 UUID.

Parameters

<i>in, out</i>	<i>buffer</i>	A buffer that is at least (UUIDLEN + 1) bytes long.
----------------	---------------	---

Return values

<i>true</i>	success
<i>false</i>	failure

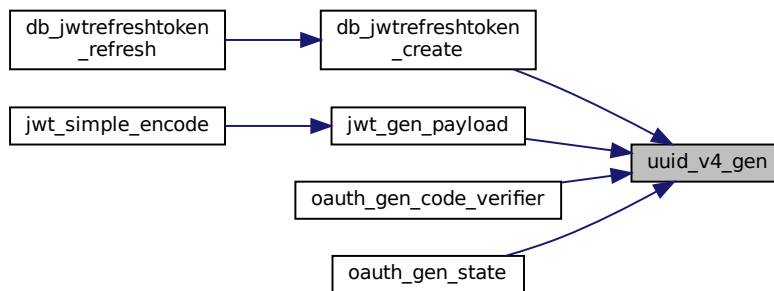
Note

Code from <https://gist.github.com/kvelakur/9069c9896577c3040030>

Definition at line 35 of file common.c.

Referenced by `db_jwtrefresh_token_create()`, `jwt_gen_payload()`, `oauth_gen_code_verifier()`, and `oauth_gen_state()`.

Here is the caller graph for this function:



5.6.2.22 writestreamtofile()

```

bool writestreamtofile (
    FILE * stream,
    const char * outpath )
  
```

Writes a stream to the file located at outpath.

The stream is read until EOF is reached.

Parameters

in	<i>stream</i>	the stream to be read from
in	<i>outpath</i>	path to the file were the output should be written to

Returns

whether the function was successful

Return values

<i>true</i>	success
<i>false</i>	An error occurred and errno might be set

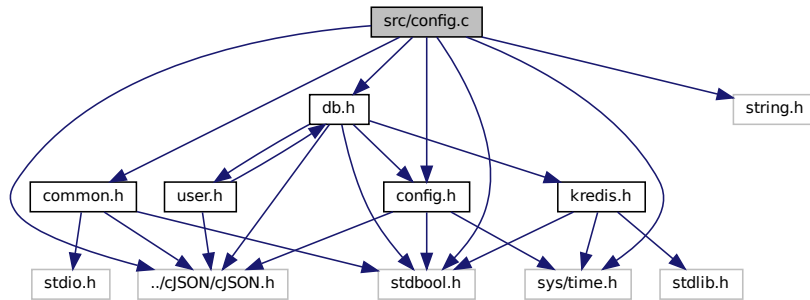
Definition at line 711 of file common.c.

5.7 src/config.c File Reference

Code for parsing the config into a native format.

```
#include "config.h"
#include "common.h"
#include "db.h"
#include "../cJSON/cJSON.h"
#include <sys/time.h>
#include <string.h>
#include <stdbool.h>
```

Include dependency graph for config.c:



Functions

- struct `config_s` * `config_parse` (char *errmsg, const char *filepath)
Parses the config.
- void `config_free` (struct `config_s` *config)
Frees a config.

5.7.1 Detailed Description

Code for parsing the config into a native format.

5.7.1.1 config.json Layout

```
* {
*   "db": { // Database settings
*     "type": "redis",
*     "ip": <string|ip of redis server|can be null>,
*     "port": <number|port of redis server>,
*     "socket": <string|path to unix socket file|can be null>,
*     "use_unix_socket": <bool|whether to use ip+port or socket for the connection>,
*     "timeout": { "seconds": <number|seconds of timeout>, "microseconds": <number|microseconds of timeout> },
*     "auth": [string|redis auth string NOTE: only pre Redis 6.0 requirepass auth string is supported]
*   },
*   "auth": { // Settings for the identity provider
*     "type": "oidc", // OpenID Connect
*     "oauth_authorize_url": <string|oauth authorize endpoint. example: https://gitlab.gwdg.de/oauth/authorize>,
*     "oauth_token_url": <string|oauth token endpoint. example: https://gitlab.gwdg.de/oauth/token>,
*     "redirect_uri": <string|url of oauthcallback endpoint. must match the redirect uri configured at the provider>,
*     "scope": <string|space separated list of oauth scopes. must be equivalent to the scope configured at the provider>,
*     "app_id": <string|the oauth application id>,
*     "app_secret": <string|the oauth application secret>,
*     "oidc_expected_issuer": <string|the expected oidc issuer. 'iss' field in id_token. example: https://gitlab.gwdg.de>
*   },
* }
```

```

*     "jwt": { // JSON Web Token settings of the JWT the user gets by the refreshlogin endpoint after authen
*         "secret":<string|JWT secret. must not be bruteforcable!>,
*         "ttl": <number|Time-To-Live of the JWT in seconds>,
*         "leeway": <number|leeway of ttl and ttl_refresh token>,
*         "ttl_refresh token": <number|Time-To-Live of the JWT Refresh Token in seconds>
*     },
*     "ssh-sign": {
*         "client_ca_private_key": <string|location of the client-ca file. example: ../../client-ca>,
*         "validity": <string|how long a signed key should be valid for. see -V option of ssh-keygen. ex
*     }
* }
*

```

5.7.2 Function Documentation

5.7.2.1 config_free()

```

void config_free (
    struct config_s * config )

```

Frees a config.

Parameters

in	<i>config</i>	the config
----	---------------	------------

Definition at line 348 of file config.c.

References `config_s::__raw`.

5.7.2.2 config_parse()

```

struct config_s* config_parse (
    char * errmsg,
    const char * filepath )

```

Parses the config.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>filepath</i>	path to the file containing the config

Returns

the parsed config

Return values

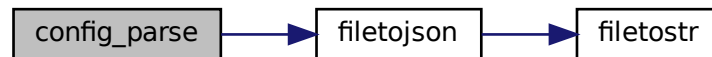
<code>NULL</code>	an error occurred and an error message has been written to <code>errmsg</code>
-------------------	--

The returned config must be freed using [`config_free\(\)`](#).

Definition at line 272 of file `config.c`.

References `config_s::__raw`, and [`filetojson\(\)`](#).

Here is the call graph for this function:



5.8 src/config.h File Reference

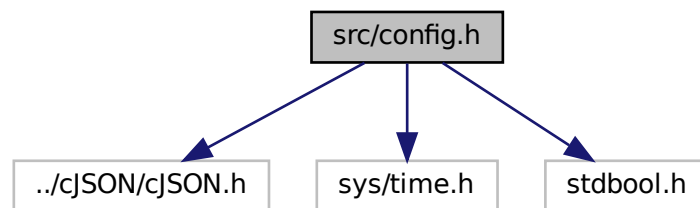
Header file for [`config.c`](#).

```

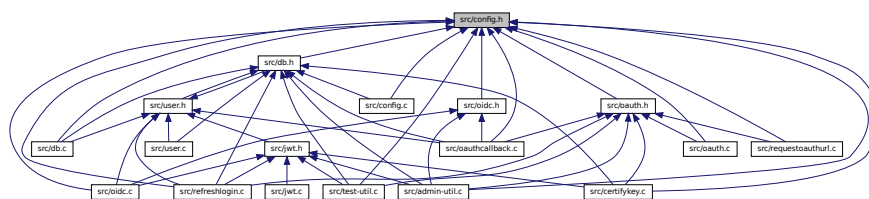
#include "../cJSON/cJSON.h"
#include <sys/time.h>
#include <stdbool.h>

```

Include dependency graph for `config.h`:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [config_db_s](#)
native representation of config.db
- struct [config_auth_s](#)
native representation of config.auth
- struct [config_jwt_s](#)
native representation of config.jwt
- struct [config_certify_s](#)
native representation of config["ssh-sign"]
- struct [config_s](#)
config struct

Functions

- struct [config_s](#) * [config_parse](#) (char *errmsg, const char *filepath)
Parses the config.
- void [config_free](#) (struct [config_s](#) *config)
Frees a config.

5.8.1 Detailed Description

Header file for [config.c](#).

5.8.2 Function Documentation

5.8.2.1 [config_free\(\)](#)

```
void config_free (  
    struct config\_s * config )
```

Frees a config.

Parameters

in	<i>config</i>	the config
----	---------------	------------

Definition at line 348 of file [config.c](#).

References [config_s::__raw](#).

5.8.2.2 config_parse()

```
struct config_s* config_parse (
    char * errmsg,
    const char * filepath )
```

Parses the config.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>filepath</i>	path to the file containing the config

Returns

the parsed config

Return values

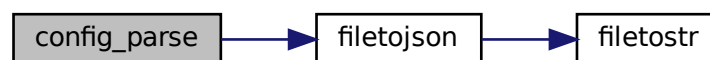
<i>NULL</i>	an error occurred and an error message has been written to errmsg
-------------	---

The returned config must be freed using [config_free\(\)](#).

Definition at line 272 of file config.c.

References `config_s::__raw`, and `filetojson()`.

Here is the call graph for this function:

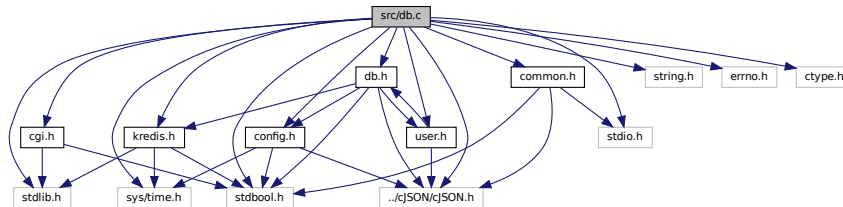


5.9 src/db.c File Reference

Code for interacting with the database on a 'higher'-level.

```
#include "db.h"
#include "common.h"
#include "cgi.h"
#include "kredis.h"
#include "config.h"
#include "user.h"
#include "../cJSON/cJSON.h"
#include <stdio.h>
```

```
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
#include <sys/time.h>
Include dependency graph for db.c:
```



Functions

- `db_t * db_connect` (char *errmsg, const struct `config_db_s` *dbconfig)
Connects to a database.
- void `db_close` (db_t *db)
Disconnects and frees a database context.
- bool `db_user_store` (char *errmsg, db_t *db, cJSON *user)
Stores a user in the database.
- cJSON * `db_user_get` (char *errmsg, db_t *db, const char *userid)
Retrieves a user from the database via their userid.
- cJSON * `db_users_get_all` (char *errmsg, db_t *db)
Retrieves all users as an array from the database.
- bool `db_jwtrefresh_token_create` (char *errmsg, db_t *db, char *token, const `user_t` *user, long ttl)
Generates a new JWT refresh token and writes it into the database.
- `user_t` * `db_jwtrefresh_token_refresh` (char *errmsg, db_t *db, char *oldtoken, char *newtoken, long ttl)
Uses and regenerates a JWT refresh token and updates the database accordingly.

5.9.1 Detailed Description

Code for interacting with the database on a 'higher'-level.

5.9.1.1 Database Layout

type	expire	key	value	comment
string	10min	jwtrefresh_token↔ :<jwtrefresh_token>	<userid>	don't delete manually. let it expire
string	never	DB_PREFIX_USE↔ R<userid>	see the JSON Syntax at db_user_store()	
set	never	DB_PREFIX_USERS	<userids>	

All keys have the extra prefix `sshca:`.

5.9.2 Function Documentation

5.9.2.1 `db_close()`

```
void db_close (  
    db_t * db )
```

Disconnects and frees a database context.

Parameters

<code>in, out</code>	<code>db</code>	the database context
----------------------	-----------------	----------------------

Definition at line 108 of file `db.c`.

References `config_s::db`, and `kredis_free()`.

Referenced by `db_connect()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.2 db_connect()

```
db_t* db_connect (
    char * errmsg,
    const struct config_db_s * dbconfig )
```

Connects to a database.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>dbconfig</i>	config for connecting to the database

Returns

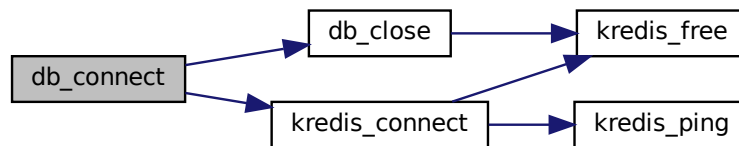
On success a database context is returned and the connection is established. Otherwise NULL is returned, errno might be set and errmsg contains an error message.

The returned pointer must be freed using [db_close\(\)](#).

Definition at line 44 of file db.c.

References [config_db_s::auth](#), [db_s::c](#), [db_close\(\)](#), [config_db_s::ip](#), [db_s::isconnected](#), [kredis_connect\(\)](#), [config_db_s::port](#), [config_db_s::socket](#), [config_db_s::timeout](#), [config_db_s::type](#), and [config_db_s::use_unix_socket](#).

Here is the call graph for this function:



5.9.2.3 db_jwtrefresh_token_create()

```
bool db_jwtrefresh_token_create (
    char * errmsg,
    db_t * db,
    char * token,
    const user_t * user,
    long ttl )
```

Generates a new JWT refresh token and writes it into the database.

JWT refresh tokens are used for authorizing JWT generation.

A token is always associated with a user. Thus when using a token you can determine for whom the JWT must be generated.

A token can only ever be used once.

A JWT refresh token itself is just a UUIDv4 string.

The database entry will expire after `ttl` seconds.

See also

[db_jwtrefresh_token_refresh\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
out	<i>token</i>	buffer for storing the JWT refresh token. size: DB_JWTREFRESHTOKEN_BUFSIZE
in	<i>user</i>	the user
in	<i>tvl</i>	how long the token should be valid in seconds

Returns

whether the function was successful.

Return values

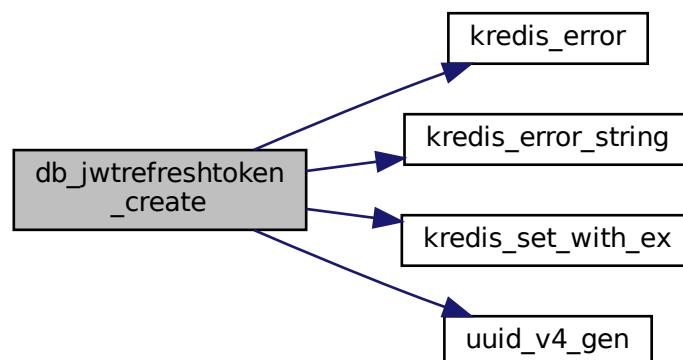
<i>true</i>	success
<i>false</i>	an error occurred and an error message has been written to errmsg

Definition at line 347 of file db.c.

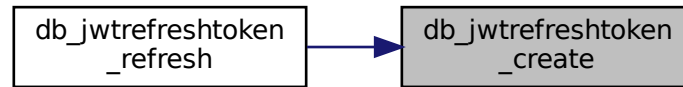
References `config_s::db`, `DB_PREFIX_JWTREFRESHTOKEN`, `user_s::id`, `kredis_error()`, `kredis_error_string()`, `kredis_set_with_ex()`, and `uuid_v4_gen()`.

Referenced by `db_jwtrefresh_token_refresh()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.4 db_jwtrefresh_token_refresh()

```

user_t* db_jwtrefresh_token_refresh (
    char * errmsg,
    db_t * db,
    char * oldtoken,
    char * newtoken,
    long ttl )
  
```

Uses and regenerates a JWT refresh token and updates the database accordingly.

This will invalidate oldtoken. newtoken should be sent back to the user.

See also

[db_jwtrefresh_token_create\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
out	<i>oldtoken</i>	buffer containing the old JWT refresh token about to be used. size: DB_JWTREFRESHTOKEN_BUFSIZE
in	<i>newtoken</i>	buffer for storing the new JWT refresh token. size: DB_JWTREFRESHTOKEN_BUFSIZE
in	<i>ttl</i>	how long the token should be valid in seconds

Note

oldtoken and newtoken may overlap.

Returns

the user associated with the token

Return values

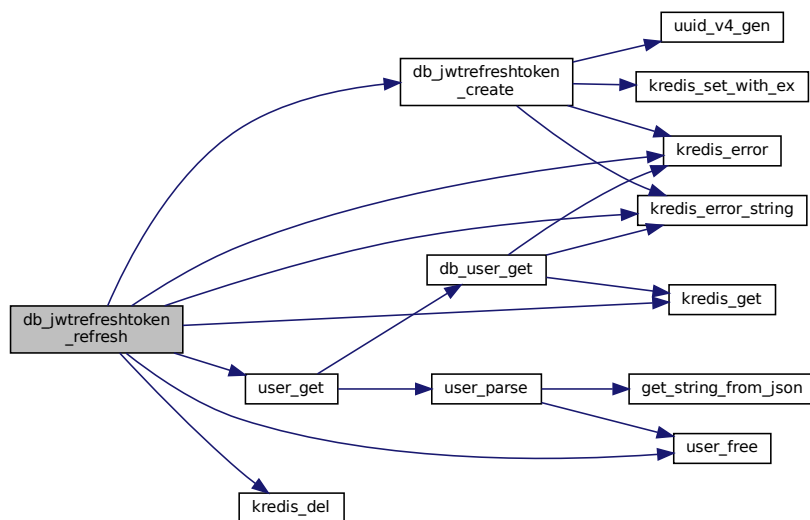
<code>NULL</code>	an error occurred or oldtoken doesn't exist in the database. An error message has been written to <code>errmsg</code> .
-------------------	---

The returned user object must be freed using [user_free\(\)](#).

Definition at line 405 of file `db.c`.

References `config_s::db`, `db_jwtrefresh_token_create()`, `DB_PREFIX_JWTREFRESHTOKEN`, `kredis_del()`, `kredis_error()`, `kredis_error_string()`, `kredis_get()`, `user_free()`, and `user_get()`.

Here is the call graph for this function:



5.9.2.5 db_user_get()

```

cJSON* db_user_get (
    char * errmsg,
    db_t * db,
    const char * userid )
  
```

Retrieves a user from the database via their `userid`.

See also

[db_user_store\(\)](#)

[user_get\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
in	<i>userid</i>	the users userid

Returns

the retrieved data as JSON

Return values

<i>NULL</i>	an error occurred or user doesn't exist in the database. An error message has been written to errmsg.
-------------	---

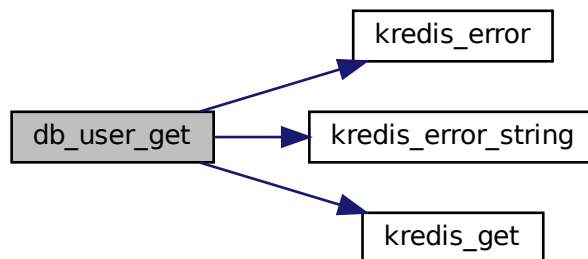
The returned cJSON object must be freed using cJSON_Delete().

Definition at line 225 of file db.c.

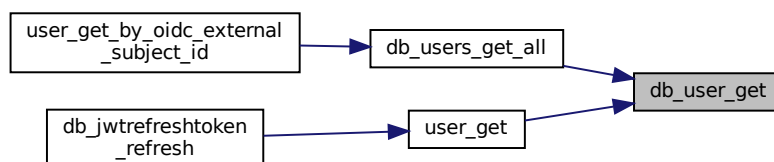
References config_s::db, DB_PREFIX_USER, kredis_error(), kredis_error_string(), and kredis_get().

Referenced by db_users_get_all(), and user_get().

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.6 db_user_store()

```
bool db_user_store (
    char * errmsg,
    db_t * db,
    cJSON * user )
```

Stores a user in the database.

The data is stored as JSON and can later be retrieved using [db_user_get\(\)](#).

The JSON syntax is as follows:

```
{
  "id": "<userid>",
  "name": "<username>",
  "oidc_external_subject_id": "<external subject id>",
  "state": "<active|blocked>",
  "principals": [
    "<principal>",
    "...",
  ]
}
```

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
in	<i>user</i>	the user

Returns

whether the function was successful.

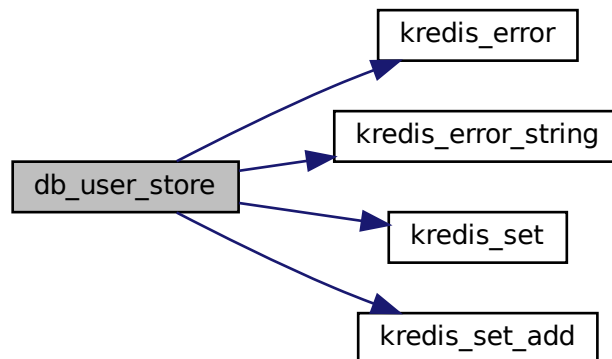
Return values

<i>true</i>	success
<i>false</i>	an error occurred and an error message has been written to errmsg

Definition at line 144 of file db.c.

References [config_s::db](#), [DB_PREFIX_USER](#), [DB_PREFIX_USERS](#), [kredis_error\(\)](#), [kredis_error_string\(\)](#), [kredis_set\(\)](#), and [kredis_set_add\(\)](#).

Here is the call graph for this function:



5.9.2.7 db_users_get_all()

```
cJSON* db_users_get_all (
    char * errmsg,
    db_t * db )
```

Retrieves all users as an array from the database.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context

Returns

a cJSON Array containing all the user objects

Return values

<i>NULL</i>	an error occurred. An error message has been written to <i>errmsg</i> .
-------------	---

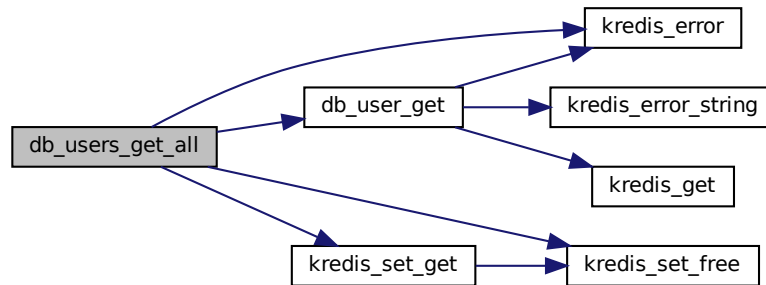
The returned array must be freed using `cJSON_Delete()`.

Definition at line 275 of file `db.c`.

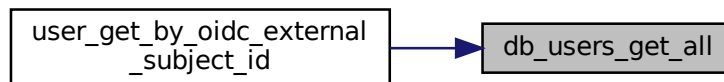
References `config_s::db`, `DB_PREFIX_USERS`, `db_user_get()`, `kredis_error()`, `kredis_set_free()`, and `kredis_set_get()`.

Referenced by `user_get_by_oidc_external_subject_id()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.10 src/db.h File Reference

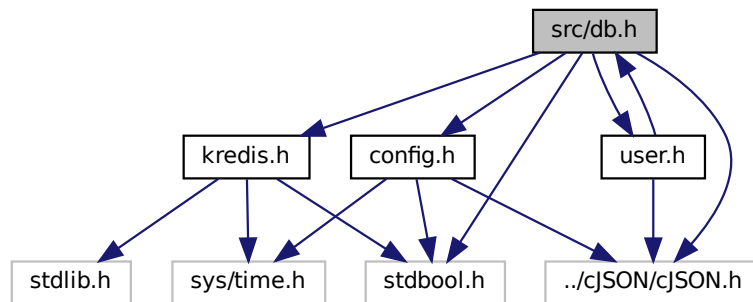
Header file for `db.c`.

```

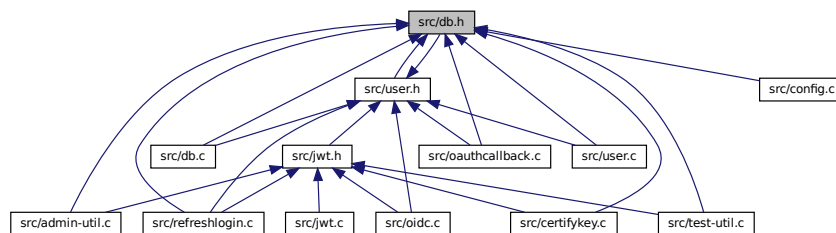
#include "kredis.h"
#include "config.h"
#include "user.h"
#include "../cJSON/cJSON.h"
#include <stdbool.h>

```


Include dependency graph for db.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `db_s`
database struct

Macros

- `#define DB_PREFIX_USER "user:"`
database prefix for user
- `#define DB_PREFIX_USERS "users:"`
database prefix for users
- `#define DB_PREFIX_JWTREFRESHTOKEN "jwtrefreshtoken:"`
database prefix for jwtrefreshtoken
- `#define DB_JWTREFRESHTOKEN_BUFSIZE (UUIDLEN + 1)`
the required buffer size for a jwtrefreshtoken

Typedefs

- typedef struct `db_s` `db_t`

Functions

- `db_t * db_connect` (char *errmsg, const struct `config_db_s` *dbconfig)
Connects to a database.
- void `db_close` (`db_t` *db)
Disconnects and frees a database context.
- bool `db_user_store` (char *errmsg, `db_t` *db, cJSON *user)
Stores a user in the database.
- cJSON * `db_user_get` (char *errmsg, `db_t` *db, const char *userid)
Retrieves a user from the database via their userid.
- cJSON * `db_users_get_all` (char *errmsg, `db_t` *db)
Retrieves all users as an array from the database.
- bool `db_jwtrefresh_token_create` (char *errmsg, `db_t` *db, char *token, const `user_t` *user, long ttl)
Generates a new JWT refresh token and writes it into the database.
- `user_t` * `db_jwtrefresh_token_refresh` (char *errmsg, `db_t` *db, char *oldtoken, char *newtoken, long ttl)
Uses and regenerates a JWT refresh token and updates the database accordingly.

5.10.1 Detailed Description

Header file for `db.c`.

5.10.2 Macro Definition Documentation

5.10.2.1 DB_PREFIX_USERS

```
#define DB_PREFIX_USERS "users:"
```

database prefix for users

TODO remove ':' (requires updating the database accordingly)

Definition at line 28 of file `db.h`.

5.10.3 Function Documentation

5.10.3.1 db_close()

```
void db_close (
    db_t * db )
```

Disconnects and frees a database context.

Parameters

in, out	<i>db</i>	the database context
---------	-----------	----------------------

Definition at line 108 of file db.c.

References `config_s::db`, and `kredis_free()`.

Referenced by `db_connect()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.3.2 db_connect()

```

db_t* db_connect (
    char * errmsg,
    const struct config_db_s * dbconfig )
  
```

Connects to a database.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>dbconfig</i>	config for connecting to the database

Returns

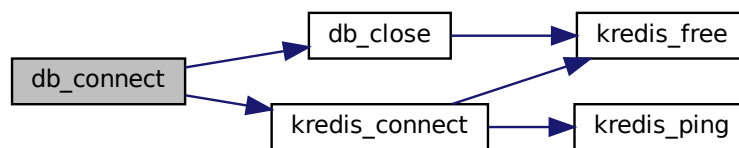
On success a database context is returned and the connection is established. Otherwise NULL is returned, `errno` might be set and `errmsg` contains an error message.

The returned pointer must be freed using [db_close\(\)](#).

Definition at line 44 of file `db.c`.

References `config_db_s::auth`, `db_s::c`, `db_close()`, `config_db_s::ip`, `db_s::isconnected`, `kredis_connect()`, `config_db_s::port`, `config_db_s::socket`, `config_db_s::timeout`, `config_db_s::type`, and `config_db_s::use_unix_socket`.

Here is the call graph for this function:

**5.10.3.3 db_jwtrefresh_token_create()**

```

bool db_jwtrefresh_token_create (
    char * errmsg,
    db_t * db,
    char * token,
    const user_t * user,
    long ttl )
  
```

Generates a new JWT refresh token and writes it into the database.

JWT refresh tokens are used for authorizing JWT generation.

A token is always associated with a user. Thus when using a token you can determine for whom the JWT must be generated.

A token can only ever be used once.

A JWT refresh token itself is just a UUIDv4 string.

The database entry will expire after `ttl` seconds.

See also

[db_jwtrefresh_token_refresh\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
out	<i>token</i>	buffer for storing the JWT refresh token. size: DB_JWTREFRESHTOKEN_BUFSIZE
in	<i>user</i>	the user
in	<i>tvl</i>	how long the token should be valid in seconds

Returns

whether the function was successful.

Return values

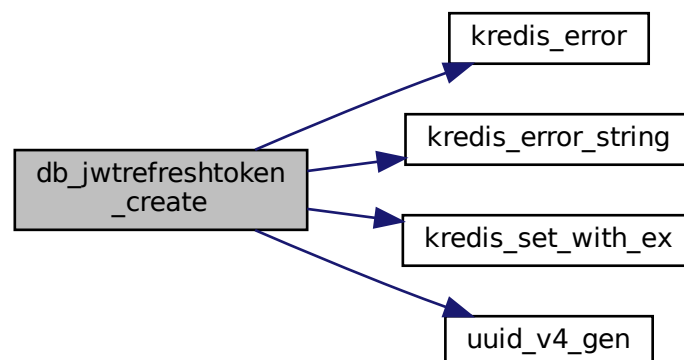
<i>true</i>	success
<i>false</i>	an error occurred and an error message has been written to errmsg

Definition at line 347 of file db.c.

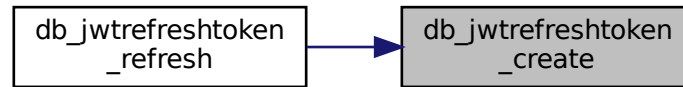
References `config_s::db`, `DB_PREFIX_JWTREFRESHTOKEN`, `user_s::id`, `kredis_error()`, `kredis_error_string()`, `kredis_set_with_ex()`, and `uuid_v4_gen()`.

Referenced by `db_jwtrefresh_token_refresh()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.3.4 db_jwtrefresh_token_refresh()

```

user_t* db_jwtrefresh_token_refresh (
    char * errmsg,
    db_t * db,
    char * oldtoken,
    char * newtoken,
    long ttl )
  
```

Uses and regenerates a JWT refresh token and updates the database accordingly.

This will invalidate oldtoken. newtoken should be sent back to the user.

See also

[db_jwtrefresh_token_create\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
out	<i>oldtoken</i>	buffer containing the old JWT refresh token about to be used. size: DB_JWTREFRESHTOKEN_BUFSIZE
in	<i>newtoken</i>	buffer for storing the new JWT refresh token. size: DB_JWTREFRESHTOKEN_BUFSIZE
in	<i>ttl</i>	how long the token should be valid in seconds

Note

oldtoken and newtoken may overlap.

Returns

the user associated with the token

Return values

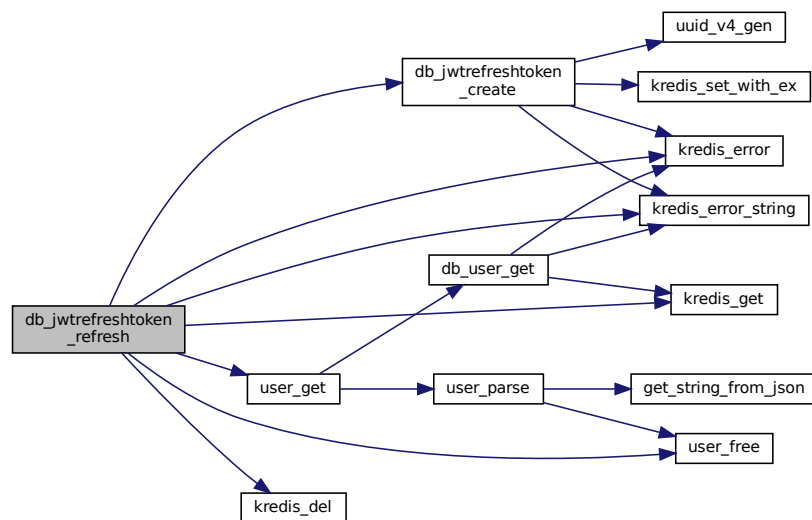
<code>NULL</code>	an error occurred or oldtoken doesn't exist in the database. An error message has been written to <code>errmsg</code> .
-------------------	---

The returned user object must be freed using [user_free\(\)](#).

Definition at line 405 of file `db.c`.

References `config_s::db`, `db_jwtrefresh_token_create()`, `DB_PREFIX_JWTREFRESHTOKEN`, `kredis_del()`, `kredis_error()`, `kredis_error_string()`, `kredis_get()`, `user_free()`, and `user_get()`.

Here is the call graph for this function:



5.10.3.5 db_user_get()

```

cJSON* db_user_get (
    char * errmsg,
    db_t * db,
    const char * userid )
  
```

Retrieves a user from the database via their `userid`.

See also

[db_user_store\(\)](#)

[user_get\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
in	<i>userid</i>	the users userid

Returns

the retrieved data as JSON

Return values

<i>NULL</i>	an error occurred or user doesn't exist in the database. An error message has been written to errmsg.
-------------	---

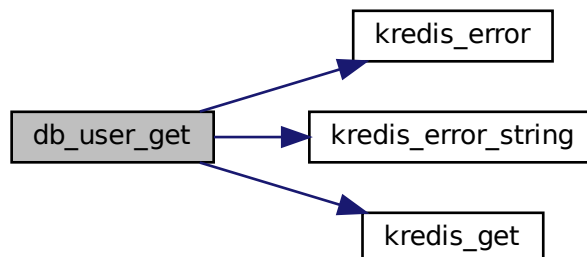
The returned cJSON object must be freed using cJSON_Delete().

Definition at line 225 of file db.c.

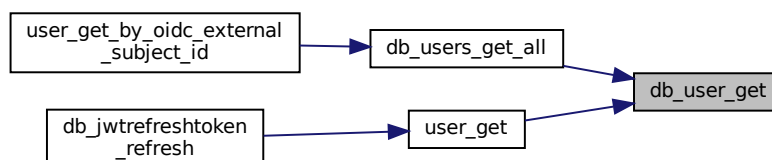
References config_s::db, DB_PREFIX_USER, kredis_error(), kredis_error_string(), and kredis_get().

Referenced by db_users_get_all(), and user_get().

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.3.6 db_user_store()

```
bool db_user_store (
    char * errmsg,
    db_t * db,
    cJSON * user )
```

Stores a user in the database.

The data is stored as JSON and can later be retrieved using [db_user_get\(\)](#).

The JSON syntax is as follows:

```
{
    "id": "<userid>",
    "name": "<username>",
    "oidc_external_subject_id": "<external subject id>",
    "state": "<active|blocked>",
    "principals": [
        "<principal>",
        "...",
    ]
}
```

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
in	<i>user</i>	the user

Returns

whether the function was successful.

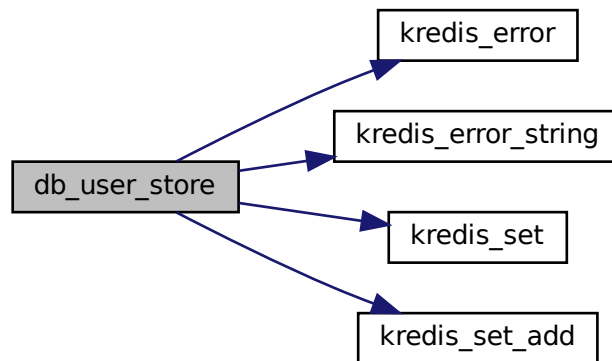
Return values

<i>true</i>	success
<i>false</i>	an error occurred and an error message has been written to <i>errmsg</i>

Definition at line 144 of file db.c.

References `config_s::db`, `DB_PREFIX_USER`, `DB_PREFIX_USERS`, `kredis_error()`, `kredis_error_string()`, `kredis_set()`, and `kredis_set_add()`.

Here is the call graph for this function:



5.10.3.7 db_users_get_all()

```

cJSON* db_users_get_all (
    char * errmsg,
    db_t * db )
  
```

Retrieves all users as an array from the database.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context

Returns

a cJSON Array containing all the user objects

Return values

<i>NULL</i>	an error occurred. An error message has been written to errmsg.
-------------	---

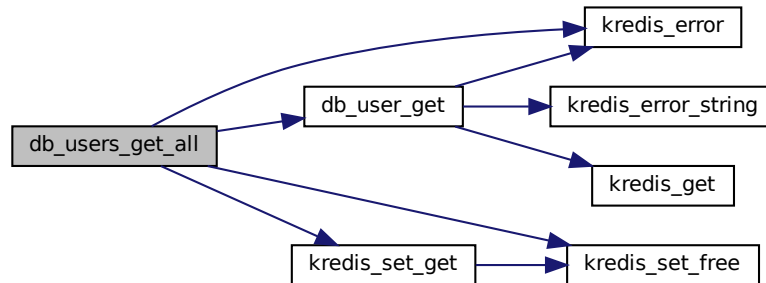
The returned array must be freed using cJSON_Delete().

Definition at line 275 of file db.c.

References config_s::db, DB_PREFIX_USERS, db_user_get(), kredis_error(), kredis_set_free(), and kredis_set_get().

Referenced by `user_get_by_oidc_external_subject_id()`.

Here is the call graph for this function:



Here is the caller graph for this function:

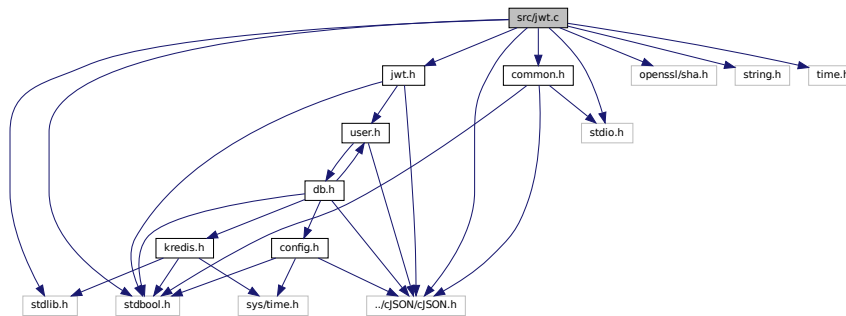


5.11 src/jwt.c File Reference

JSON Web Token.

```
#include "jwt.h"  
#include "common.h"  
#include "../cJSON/cJSON.h"  
#include <openssl/sha.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <stdbool.h>  
#include <string.h>  
#include <time.h>
```

Include dependency graph for jwt.c:



Functions

- char * [jwt_simple_encode](#) (char *errmsg, const [user_t](#) *user, unsigned long expin, unsigned long leeway, const char *key)
Wrapper function for creating, encoding and signing a JWT.
- char * [jwt_simple_decode](#) (char *errmsg, const char *jwtstr, const char *key)
Wrapper function for verifying and decoding a JWT and extracting the userid from the sub claim.
- cJSON * [jwt_gen_payload](#) (char *errmsg, const [user_t](#) *user, unsigned long expin, unsigned long leeway)
Generates a JWT payload.
- char * [jwt_encode_and_sign](#) (const cJSON *payload, const char *key)
Encodes and signs a JWT.
- bool [jwt_verify_encoding](#) (const char *jwtstr)
Verifies the encoding of a JWT string.
- bool [jwt_verify_signature](#) (const char *jwtstr, const char *key)
Verifies the signature of a JWT string.
- cJSON * [jwt_decode](#) (const char *jwtstr)
Extracts the payload of a JWT string.
- bool [jwt_has_expired](#) (const cJSON *payload)
Tests whether a JWT payload has expired.

5.11.1 Detailed Description

JSON Web Token.

This implementation of JWTs is NOT 100% conforming to RFC7519! See <https://datatracker.ietf.org/doc/html/rfc7519#section-8> . For example the algorithm type "none" is NOT implemented! Also the validation is NOT 100% conforming to <https://datatracker.ietf.org/doc/html/rfc7519#section-7.2> , as this implementation assumes the algorithm and thus can and does check the signature first.

@TODO Make this implementation 100% conforming to [RFC7519](#).

5.11.2 Function Documentation

5.11.2.1 jwt_decode()

```
cJSON* jwt_decode (
    const char * jwtstr )
```

Extracts the payload of a JWT string.

Parameters

in	<i>jwtstr</i>	the JWT as a string
----	---------------	---------------------

Returns

the payload as JSON

Return values

<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

Note

You should call [jwt_verify_signature\(\)](#) before calling this function.

You should call [jwt_has_expired\(\)](#) after calling this function.

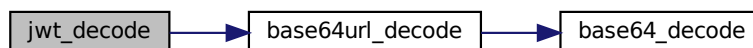
The returned cJSON object must be freed using `cJSON_Delete()`.

Definition at line 465 of file `jwt.c`.

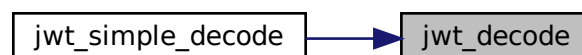
References [base64url_decode\(\)](#), and `JWT_HEADER_TYP`.

Referenced by [jwt_simple_decode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.2.2 `jwt_encode_and_sign()`

```
char* jwt_encode_and_sign (
    const cJSON * payload,
    const char * key )
```

Encodes and signs a JWT.

Uses HS256 as algorithm.

The payload can be generated using [jwt_gen_payload\(\)](#).

Note

The payload is taken as is and NOT further modified by this function.

Parameters

in	<i>payload</i>	the JWT payload
in	<i>key</i>	the key/secret

Returns

a signed JWT as a string

Return values

<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

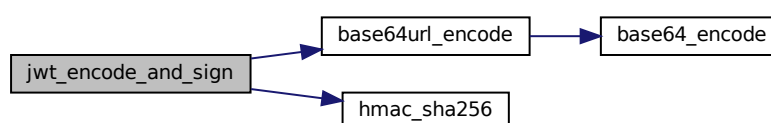
The returned string must be freed using `free()`.

Definition at line 283 of file `jwt.c`.

References `base64url_encode()`, `hmac_sha256()`, `JWT_HEADER_ALG`, and `JWT_HEADER_TYP`.

Referenced by `jwt_simple_encode()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.2.3 jwt_gen_payload()

```

cJSON* jwt_gen_payload (
    char * errmsg,
    const user_t * user,
    unsigned long expin,
    unsigned long leeway )
  
```

Generates a JWT payload.

The payload contains the following claims: iss: JWT_CLAIM_ISS sub: userid iat: current time as nbf: iat - leeway exp: iat + expin + leeway jti: UUIDv4 name: username

See also

<https://datatracker.ietf.org/doc/html/rfc7519#section-4.1>

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>user</i>	the user
in	<i>expin</i>	number of seconds for which the JWT is valid
in	<i>leeway</i>	leeway in seconds to account for clock skew (see https://datatracker.ietf.org/doc/html/rfc7519#section-4.1.4)

Returns

the payload as JSON

Return values

<i>NULL</i>	an error occurred and errno might be set. An error message has been written to errmsg.
-------------	--

The returned cJSON object must be freed using cJSON_Delete().

Definition at line 168 of file jwt.c.

References `user_s::id`, `JWT_CLAIM_ISS`, `user_s::name`, `uuid_v4_gen()`, and `UUIDLEN`.

Referenced by `jwt_simple_encode()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.2.4 `jwt_has_expired()`

```
bool jwt_has_expired (  
    const cJSON * payload )
```

Tests whether a JWT payload has expired.

This function compares the `nbf` and `exp` claims in the payload to the current time.

`nbf` is ignored, if `nbf` doesn't exist in the payload.

Parameters

<code>in</code>	<code>payload</code>	the JWT payload
-----------------	----------------------	-----------------

Returns

whether the payload has expired

Return values

<i>false</i>	the payload has NOT expired
<i>true</i>	the payload has expired and is thus invalid

Definition at line 566 of file jwt.c.

Referenced by `jwt_simple_decode()`.

Here is the caller graph for this function:

5.11.2.5 `jwt_simple_decode()`

```

char* jwt_simple_decode (
    char * errmsg,
    const char * jwtstr,
    const char * key )
  
```

Wrapper function for verifying and decoding a JWT and extracting the userid from the sub claim.

This function calls:

1. [jwt_verify_encoding\(\)](#)
2. [jwt_verify_signature\(\)](#)
3. [jwt_decode\(\)](#)
4. [jwt_has_expired\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>jwtstr</i>	the JWT as a string
in	<i>key</i>	the key/secret

Returns

the sub claim of the payload of the JWT. This should be a userid.

Return values

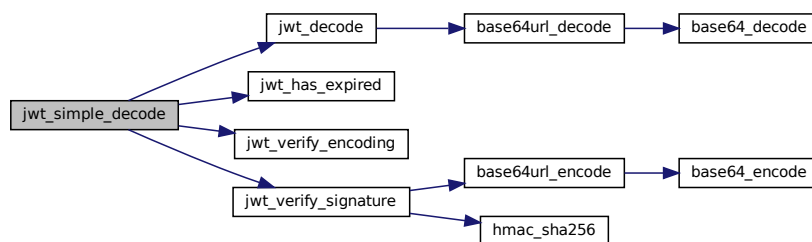
<i>NULL</i>	an error occurred and <i>errno</i> might be set, or <i>jwtstr</i> failed verification. An error message has been written to <i>errmsg</i> .
-------------	---

The returned string must be freed using `free()`.

Definition at line 90 of file `jwt.c`.

References `jwt_decode()`, `jwt_has_expired()`, `jwt_verify_encoding()`, and `jwt_verify_signature()`.

Here is the call graph for this function:



5.11.2.6 `jwt_simple_encode()`

```

char* jwt_simple_encode (
    char * errmsg,
    const user_t * user,
    unsigned long expin,
    unsigned long leeway,
    const char * key )
  
```

Wrapper function for creating, encoding and signing a JWT.

This function calls:

1. [jwt_gen_payload\(\)](#)
2. [jwt_encode_and_sign\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>user</i>	the user
in	<i>expin</i>	number of seconds for which the JWT is valid
in	<i>leeway</i>	leeway in seconds to account for clock skew (see https://datatracker.ietf.org/doc/html/rfc7519#section-4.1.4)
in	<i>key</i>	the key/secret

Returns

a signed JWT as a string

Return values

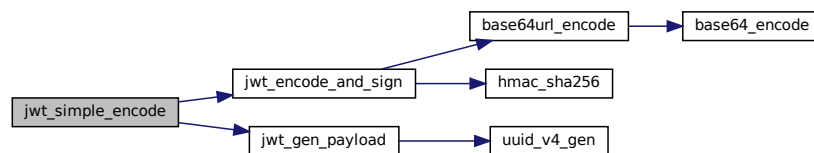
<i>NULL</i>	an error occurred and <code>errno</code> might be set. An error message has been written to <code>errmsg</code> .
-------------	---

The returned string must be freed using `free()`.

Definition at line 45 of file `jwt.c`.

References `jwt_encode_and_sign()`, and `jwt_gen_payload()`.

Here is the call graph for this function:

**5.11.2.7 jwt_verify_encoding()**

```
bool jwt_verify_encoding (
    const char * jwtstr )
```

Verifies the encoding of a JWT string.

This function is a really basic encoding verification. It only checks whether all characters are valid `base64url` characters, the number of dots and the minimal distance between the dots. That's it!

Parameters

in	<i>jwtstr</i>	the JWT as a string
----	---------------	---------------------

Returns

whether `jwtstr` is encoded correctly

Return values

<i>true</i>	<code>jwtstr</code> might be a valid JWT string
<i>false</i>	<code>jwtstr</code> is not a valid JWT string

Definition at line 363 of file jwt.c.

Referenced by `jwt_simple_decode()`.

Here is the caller graph for this function:



5.11.2.8 `jwt_verify_signature()`

```

bool jwt_verify_signature (
    const char * jwtstr,
    const char * key )
  
```

Verifies the signature of a JWT string.

Parameters

in	<i>jwtstr</i>	the JWT as a string
in	<i>key</i>	the key to be used for verification

Returns

whether *jwtstr* is signed by *key*

Return values

<i>true</i>	<i>jwtstr</i> was signed with <i>key</i>
<i>false</i>	<i>jwtstr</i> was NOT signed with <i>key</i>

Note

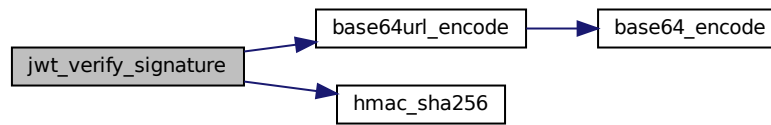
You should call [jwt_verify_encoding\(\)](#) before calling this function.

Definition at line 408 of file jwt.c.

References `base64url_encode()`, and `hmac_sha256()`.

Referenced by `jwt_simple_decode()`.

Here is the call graph for this function:



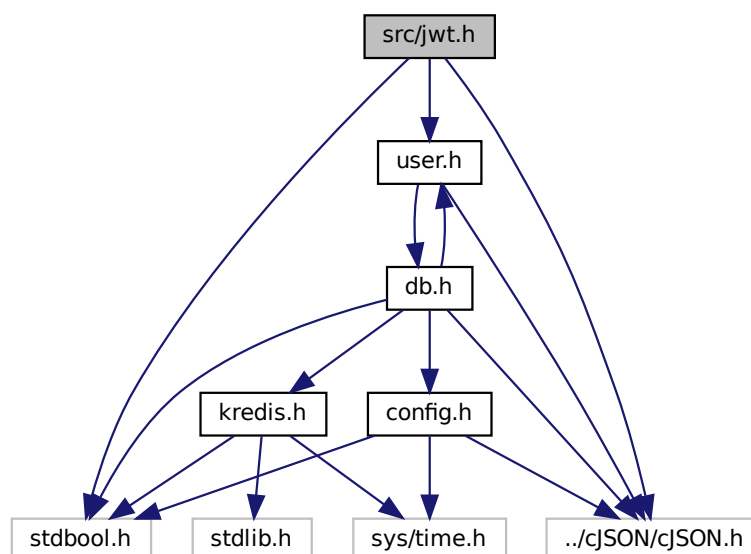
Here is the caller graph for this function:



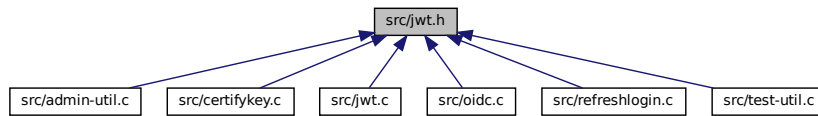
5.12 src/jwt.h File Reference

Header file for [jwt.c](#).

```
#include "user.h"  
#include "../cJSON/cJSON.h"  
#include <stdbool.h>  
Include dependency graph for jwt.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- `#define JWT_HEADER_TYP "JWT"`
JWT Type.
- `#define JWT_HEADER_ALG "HS256"`
used JWT Algorithm
- `#define JWT_CLAIM_ISS "SSHCA"`
JWT Issuer.

Functions

- `char * jwt_simple_encode` (`char *errmsg`, `const user_t *user`, `unsigned long expin`, `unsigned long leeway`, `const char *key`)
Wrapper function for creating, encoding and signing a JWT.
- `char * jwt_simple_decode` (`char *errmsg`, `const char *jwtstr`, `const char *key`)
Wrapper function for verifying and decoding a JWT and extracting the userid from the sub claim.
- `cJSON * jwt_gen_payload` (`char *errmsg`, `const user_t *user`, `unsigned long expin`, `unsigned long leeway`)
Generates a JWT payload.
- `char * jwt_encode_and_sign` (`const cJSON *payload`, `const char *key`)
Encodes and signs a JWT.
- `bool jwt_verify_encoding` (`const char *jwtstr`)
Verifies the encoding of a JWT string.
- `bool jwt_verify_signature` (`const char *jwtstr`, `const char *key`)
Verifies the signature of a JWT string.
- `cJSON * jwt_decode` (`const char *jwtstr`)
Extracts the payload of a JWT string.
- `bool jwt_has_expired` (`const cJSON *payload`)
Tests whether a JWT payload has expired.

5.12.1 Detailed Description

Header file for [jwt.c](#).

5.12.2 Function Documentation

5.12.2.1 `jwt_decode()`

```
cJSON* jwt_decode (
    const char * jwtstr )
```

Extracts the payload of a JWT string.

Parameters

in	<i>jwtstr</i>	the JWT as a string
----	---------------	---------------------

Returns

the payload as JSON

Return values

<i>NULL</i>	an error occurred and <i>errno</i> might be set
-------------	---

Note

You should call [jwt_verify_signature\(\)](#) before calling this function.

You should call [jwt_has_expired\(\)](#) after calling this function.

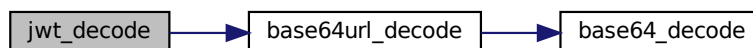
The returned cJSON object must be freed using `cJSON_Delete()`.

Definition at line 465 of file `jwt.c`.

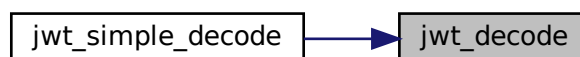
References [base64url_decode\(\)](#), and `JWT_HEADER_TYP`.

Referenced by [jwt_simple_decode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.2.2 `jwt_encode_and_sign()`

```
char* jwt_encode_and_sign (
    const cJSON * payload,
    const char * key )
```

Encodes and signs a JWT.

Uses HS256 as algorithm.

The payload can be generated using [jwt_gen_payload\(\)](#).

Note

The payload is taken as is and NOT further modified by this function.

Parameters

in	<i>payload</i>	the JWT payload
in	<i>key</i>	the key/secret

Returns

a signed JWT as a string

Return values

<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

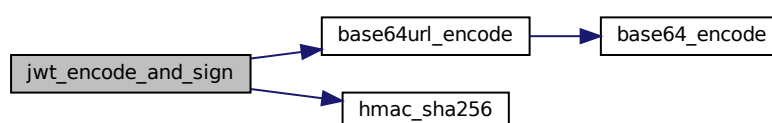
The returned string must be freed using `free()`.

Definition at line 283 of file `jwt.c`.

References `base64url_encode()`, `hmac_sha256()`, `JWT_HEADER_ALG`, and `JWT_HEADER_TYP`.

Referenced by `jwt_simple_encode()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.2.3 jwt_gen_payload()

```

cJSON* jwt_gen_payload (
    char * errmsg,
    const user_t * user,
    unsigned long expin,
    unsigned long leeway )
  
```

Generates a JWT payload.

The payload contains the following claims: iss: JWT_CLAIM_ISS sub: userid iat: current time as nbf: iat - leeway exp: iat + expin + leeway jti: UUIDv4 name: username

See also

<https://datatracker.ietf.org/doc/html/rfc7519#section-4.1>

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>user</i>	the user
in	<i>expin</i>	number of seconds for which the JWT is valid
in	<i>leeway</i>	leeway in seconds to account for clock skew (see https://datatracker.ietf.org/doc/html/rfc7519#section-4.1.4)

Returns

the payload as JSON

Return values

<i>NULL</i>	an error occurred and errno might be set. An error message has been written to errmsg.
-------------	--

The returned cJSON object must be freed using cJSON_Delete().

Definition at line 168 of file jwt.c.

References `user_s::id`, `JWT_CLAIM_ISS`, `user_s::name`, `uuid_v4_gen()`, and `UUIDLEN`.

Referenced by `jwt_simple_encode()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.2.4 `jwt_has_expired()`

```
bool jwt_has_expired (
    const cJSON * payload )
```

Tests whether a JWT payload has expired.

This function compares the `nbf` and `exp` claims in the payload to the current time.

`nbf` is ignored, if `nbf` doesn't exist in the payload.

Parameters

<code>in</code>	<code>payload</code>	the JWT payload
-----------------	----------------------	-----------------

Returns

whether the payload has expired

Return values

<i>false</i>	the payload has NOT expired
<i>true</i>	the payload has expired and is thus invalid

Definition at line 566 of file jwt.c.

Referenced by `jwt_simple_decode()`.

Here is the caller graph for this function:

5.12.2.5 `jwt_simple_decode()`

```

char* jwt_simple_decode (
    char * errmsg,
    const char * jwtstr,
    const char * key )
  
```

Wrapper function for verifying and decoding a JWT and extracting the userid from the sub claim.

This function calls:

1. [jwt_verify_encoding\(\)](#)
2. [jwt_verify_signature\(\)](#)
3. [jwt_decode\(\)](#)
4. [jwt_has_expired\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>jwtstr</i>	the JWT as a string
in	<i>key</i>	the key/secret

Returns

the sub claim of the payload of the JWT. This should be a userid.

Return values

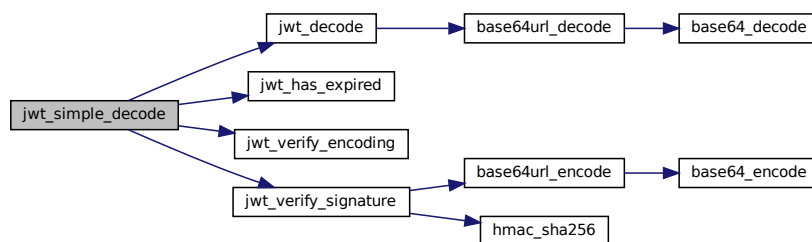
<i>NULL</i>	an error occurred and <code>errno</code> might be set, or <code>jwtstr</code> failed verification. An error message has been written to <code>errmsg</code> .
-------------	---

The returned string must be freed using `free()`.

Definition at line 90 of file `jwt.c`.

References `jwt_decode()`, `jwt_has_expired()`, `jwt_verify_encoding()`, and `jwt_verify_signature()`.

Here is the call graph for this function:



5.12.2.6 jwt_simple_encode()

```

char* jwt_simple_encode (
    char * errmsg,
    const user_t * user,
    unsigned long expin,
    unsigned long leeway,
    const char * key )
  
```

Wrapper function for creating, encoding and signing a JWT.

This function calls:

1. [jwt_gen_payload\(\)](#)
2. [jwt_encode_and_sign\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>user</i>	the user
in	<i>expin</i>	number of seconds for which the JWT is valid
in	<i>leeway</i>	leeway in seconds to account for clock skew (see https://datatracker.ietf.org/doc/html/rfc7519#section-4.1.4)
in	<i>key</i>	the key/secret

Returns

a signed JWT as a string

Return values

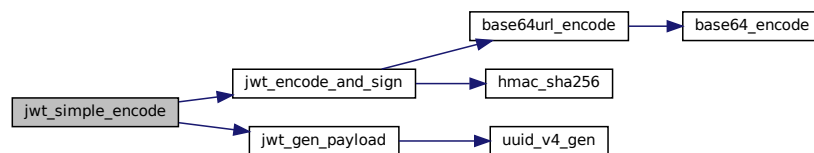
<i>NULL</i>	an error occurred and <code>errno</code> might be set. An error message has been written to <code>errmsg</code> .
-------------	---

The returned string must be freed using `free()`.

Definition at line 45 of file `jwt.c`.

References `jwt_encode_and_sign()`, and `jwt_gen_payload()`.

Here is the call graph for this function:

**5.12.2.7 jwt_verify_encoding()**

```

bool jwt_verify_encoding (
    const char * jwtstr )
  
```

Verifies the encoding of a JWT string.

This function is a really basic encoding verification. It only checks whether all characters are valid `base64url` characters, the number of dots and the minimal distance between the dots. That's it!

Parameters

in	<i>jwtstr</i>	the JWT as a string
----	---------------	---------------------

Returns

whether `jwtstr` is encoded correctly

Return values

<i>true</i>	<code>jwtstr</code> might be a valid JWT string
<i>false</i>	<code>jwtstr</code> is not a valid JWT string

Definition at line 363 of file jwt.c.

Referenced by `jwt_simple_decode()`.

Here is the caller graph for this function:



5.12.2.8 `jwt_verify_signature()`

```

bool jwt_verify_signature (
    const char * jwtstr,
    const char * key )
  
```

Verifies the signature of a JWT string.

Parameters

in	<i>jwtstr</i>	the JWT as a string
in	<i>key</i>	the key to be used for verification

Returns

whether `jwtstr` is signed by `key`

Return values

<i>true</i>	<code>jwtstr</code> was signed with <code>key</code>
<i>false</i>	<code>jwtstr</code> was NOT signed with <code>key</code>

Note

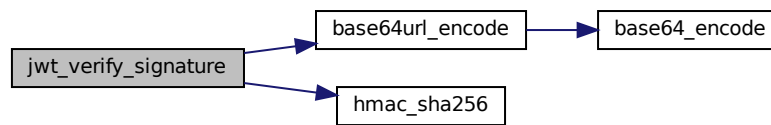
You should call `jwt_verify_encoding()` before calling this function.

Definition at line 408 of file jwt.c.

References `base64url_encode()`, and `hmac_sha256()`.

Referenced by `jwt_simple_decode()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.13 src/kredis.c File Reference

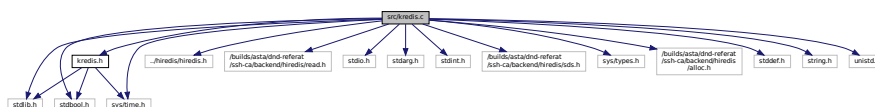
Simple ABI for hiredis.

```

#include "kredis.h"
#include "../hiredis/hiredis.h"
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>

```

Include dependency graph for `kredis.c`:



Data Structures

- struct [kredis_s](#)

Functions

- void `kredis_free` (`kredis_t *k`)
Disconnects and frees a `kredis_t` object.
- `kredis_t * kredis_connect` (`const char *ip`, `int port`, `bool is_unix_socket`, `struct timeval timeout`, `const char *auth`)
Connects to a redis server.
- bool `kredis_ping` (`kredis_t *k`)
Ping the redis server.
- int `kredis_error` (`kredis_t *k`)
Gets the current error code as an integer.
- `const char * kredis_error_string` (`kredis_t *k`)
Gets the current error code as a string.
- `char * kredis_get` (`kredis_t *k`, `const char *key`)
Native Redis GET Command.
- bool `kredis_set` (`kredis_t *k`, `const char *key`, `const char *value`)
Native Redis SET Command.
- bool `kredis_set_with_ex` (`kredis_t *k`, `const char *key`, `const char *value`, `long ex`)
Native Redis SET Command with EX set.
- bool `kredis_del` (`kredis_t *k`, `const char *key`)
Native Redis DEL Command.
- bool `kredis_set_add` (`kredis_t *k`, `const char *key`, `const char *member`)
Native Redis SADD Command.
- long long `kredis_set_num` (`kredis_t *k`, `const char *key`)
Native Redis SCARD Command.
- bool `kredis_set_ismember` (`kredis_t *k`, `const char *key`, `const char *member`)
Native Redis SISMEMBER Command.
- bool `kredis_set_rem` (`kredis_t *k`, `const char *key`, `const char *member`)
Native Redis SREM Command.
- `char ** kredis_set_get` (`kredis_t *k`, `const char *key`, `size_t *nummembers`)
Native Redis SMEMBERS Command.
- void `kredis_set_free` (`char **elements`, `size_t nummembers`)
Frees the returned array from `kredis_set_get()`
- bool `kredis_hash_set` (`kredis_t *k`, `const char *key`, `const char *field`, `const char *value`)
Native Redis HSET Command.
- bool `kredis_hash_exists` (`kredis_t *k`, `const char *key`, `const char *field`)
Native Redis HEXISTS Command.
- `char * kredis_hash_get` (`kredis_t *k`, `const char *key`, `const char *field`)
Native Redis HGET Command.

5.13.1 Detailed Description

Simple ABI for hiredis.

See also

<https://github.com/redis/hiredis>

Note

The redis server must be at least version 2.6.12.

5.13.2 Function Documentation

5.13.2.1 kredis_connect()

```
kredis_t* kredis_connect (
    const char * ip,
    int port,
    bool is_unix_socket,
    struct timeval timeout,
    const char * auth )
```

Connects to a redis server.

Parameters

in	<i>ip</i>	The IP Address of the server or the path to a unix socket. If NULL set by default to 127.0.0.1.
in	<i>port</i>	The port of the redis server. Note: The default redis port is 6379.
in	<i>is_unix_socket</i>	Whether ip is a path to a unix socket. port is ignored if set.
in	<i>timeout</i>	Sets the connection timeout.
in	<i>auth</i>	The redis server password specified in the server config by requirepass. Ignored if equal to NULL.

Returns

On success a kredis context is returned. Otherwise NULL is returned and errno might be set.

The returned kredis context must be freed using [kredis_free\(\)](#).

Note

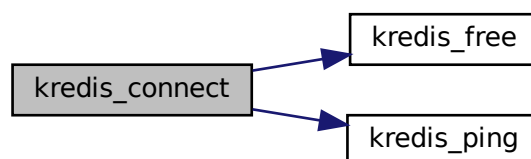
Only the old authentication method prior to Redis 6.0 is supported.

Definition at line 81 of file kredis.c.

References [kredis_free\(\)](#), [kredis_ping\(\)](#), and [kredis_s::rc](#).

Referenced by [db_connect\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.2.2 kredis_del()

```

bool kredis_del (
    kredis_t * k,
    const char * key )
  
```

Native Redis DEL Command.

Removes the specified keys. A key is ignored if it does not exist.

See also

<https://redis.io/commands/del>

Parameters

<i>in, out</i>	<i>k</i>	the kredis context
<i>in</i>	<i>key</i>	the key

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

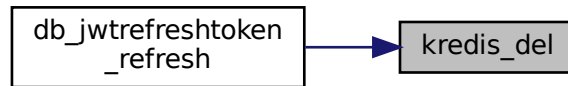
Check for errors using `kredis_error()` after calling this function.

Definition at line 383 of file kredis.c.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

Referenced by `db_jwtrefresh_token_refresh()`.

Here is the caller graph for this function:



5.13.2.3 `kredis_error()`

```
int kredis_error (  
    kredis_t * k )
```

Gets the current error code as an integer.

In case an error actually happend, you can use `kredis_error_string()` to get the error message as a string.

Parameters

in	<i>k</i>	the kredis context
----	----------	--------------------

Returns

the error code as an integer

Return values

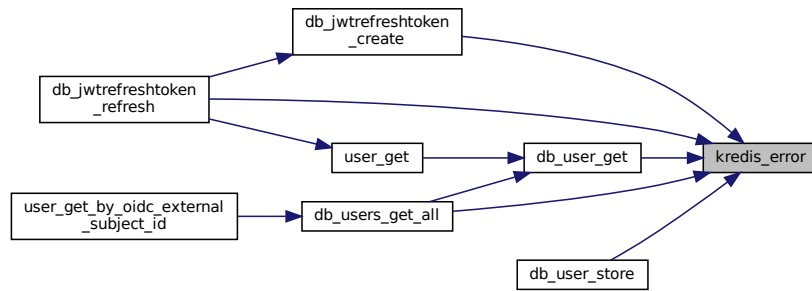
0	indicates no error
---	--------------------

Definition at line 192 of file `kredis.c`.

References `kredis_s::rc`.

Referenced by `db_jwtrefresh_token_create()`, `db_jwtrefresh_token_refresh()`, `db_user_get()`, `db_user_store()`, and `db_users_get_all()`.

Here is the caller graph for this function:



5.13.2.4 kredis_error_string()

```
const char* kredis_error_string (
    kredis_t * k )
```

Gets the current error code as a string.

This function cannot be used for checking whether an error actually occurred, use [kredis_error\(\)](#) instead.

An empty string is returned if no error happend.

Parameters

in	<i>k</i>	the kredis context
----	----------	--------------------

Returns

the error message

Return values

<i>NULL</i>	invalid argument or invalid redisContext
-------------	--

Note

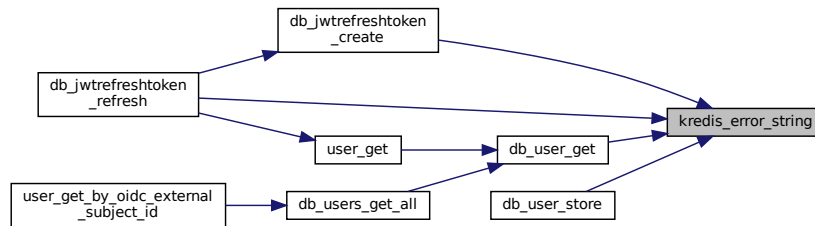
The returned string **MUST NOT** be freed.

Definition at line 213 of file kredis.c.

References [kredis_s::rc](#).

Referenced by [db_jwtrefresh_token_create\(\)](#), [db_jwtrefresh_token_refresh\(\)](#), [db_user_get\(\)](#), and [db_user_store\(\)](#).

Here is the caller graph for this function:



5.13.2.5 kredis_free()

```
void kredis_free (
    kredis_t * k )
```

Disconnects and frees a kredis_t object.

Parameters

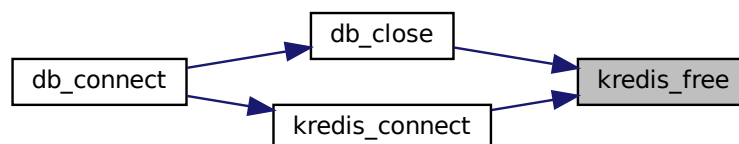
in, out	<i>k</i>	the kredis context
---------	----------	--------------------

Definition at line 57 of file kredis.c.

References kredis_s::rc.

Referenced by db_close(), and kredis_connect().

Here is the caller graph for this function:



5.13.2.6 kredis_get()

```
char* kredis_get (
    kredis_t * k,
    const char * key )
```

Native Redis GET Command.

Get the value of key.

The returned string must be freed using free().

See also

<https://redis.io/commands/get>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key

Returns

the value of key

Return values

<i>NULL</i>	key doesn't exists or isn't a string
-------------	--------------------------------------

Note

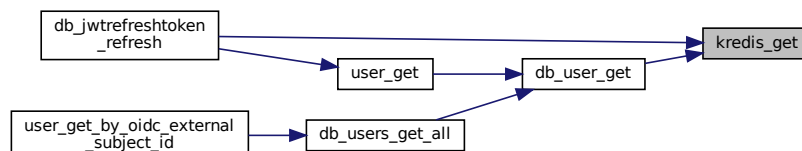
Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 238 of file kredis.c.

References KREDIS_PREFIX, and kredis_s::rc.

Referenced by db_jwtrefresh_token_refresh(), and db_user_get().

Here is the caller graph for this function:



5.13.2.7 kredis_hash_exists()

```
bool kredis_hash_exists (
    kredis_t * k,
    const char * key,
    const char * field )
```

Native Redis HEXISTS Command.

Returns if field is an existing field in the hash stored at key.

See also

<https://redis.io/commands/hexists>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>field</i>	the field

Returns

whether the field exists in the hash

Return values

<i>true</i>	the hash contains field
<i>false</i>	the hash does not contain field, key does not exist, or other failure

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 719 of file kredis.c.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

5.13.2.8 kredis_hash_get()

```
char* kredis_hash_get (
    kredis_t * k,
    const char * key,
    const char * field )
```

Native Redis HGET Command.

Returns the value associated with field in the hash stored at key.

The returned string must be freed using `free()`.

See also

<https://redis.io/commands/hget>

Parameters

<code>in, out</code>	<code>k</code>	the kredis context
<code>in</code>	<code>key</code>	the key
<code>in</code>	<code>field</code>	the field

Returns

the value

Return values

<code>NULL</code>	something failed, errno might be set
-------------------	--------------------------------------

Note

Check for errors using `kredis_error()` after calling this function.

Definition at line 759 of file `kredis.c`.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

5.13.2.9 kredis_hash_set()

```
bool kredis_hash_set (
    kredis_t * k,
    const char * key,
    const char * field,
    const char * value )
```

Native Redis HSET Command.

Sets `field` in the hash stored at `key` to `value`. If `key` does not exist, a new key holding a hash is created. If `field` already exists in the hash, it is overwritten.

See also

<https://redis.io/commands/hset>

Parameters

<code>in, out</code>	<code>k</code>	the kredis context
<code>in</code>	<code>key</code>	the key
<code>in</code>	<code>field</code>	the field
<code>in</code>	<code>value</code>	the value

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 681 of file kredis.c.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

5.13.2.10 kredis_ping()

```
bool kredis_ping (  
    kredis_t * k )
```

Ping the redis server.

Parameters

<i>in, out</i>	<i>k</i>	the kredis context
----------------	----------	--------------------

Returns

whether the ping was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

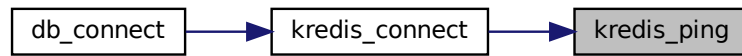
Check for errors using [kredis_error\(\)](#) on failure.

Definition at line 160 of file kredis.c.

References `kredis_s::rc`.

Referenced by `kredis_connect()`.

Here is the caller graph for this function:



5.13.2.11 kredis_set()

```

bool kredis_set (
    kredis_t * k,
    const char * key,
    const char * value )
  
```

Native Redis SET Command.

Set key to hold the string value. If key already holds a value, it is overwritten, regardless of its type. Any previous time to live associated with the key is discarded on successful SET operation.

See also

<https://redis.io/commands/set>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>value</i>	the value

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

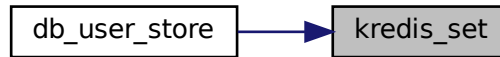
Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 284 of file kredis.c.

References KREDIS_PREFIX, and kredis_s::rc.

Referenced by db_user_store().

Here is the caller graph for this function:



5.13.2.12 kredis_set_add()

```

bool kredis_set_add (
    kredis_t * k,
    const char * key,
    const char * member )
  
```

Native Redis SADD Command.

Add the specified member to the set stored at key. Specified members that are already a member of this set are ignored. If key does not exist, a new set is created before adding the specified member.

An error occurs when the value stored at key is not a set.

See also

<https://redis.io/commands/sadd>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>member</i>	the member to be added

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 426 of file `kredis.c`.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

Referenced by `db_user_store()`.

Here is the caller graph for this function:

**5.13.2.13 kredis_set_free()**

```

void kredis_set_free (
    char ** elements,
    size_t nummembers )
  
```

Frees the returned array from [kredis_set_get\(\)](#)

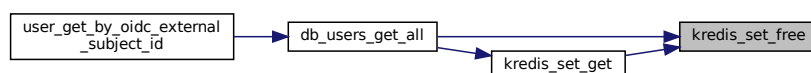
Parameters

in, out	<i>elements</i>	the array
in	<i>nummembers</i>	the size of the array

Definition at line 650 of file `kredis.c`.

Referenced by `db_users_get_all()`, and `kredis_set_get()`.

Here is the caller graph for this function:



5.13.2.14 kredis_set_get()

```
char** kredis_set_get (
    kredis_t * k,
    const char * key,
    size_t * nummembers )
```

Native Redis SMEMBERS Command.

Returns all the members of the set value stored at key.

The returned array must be freed using [kredis_set_free\(\)](#).

See also

<https://redis.io/commands/smembers>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
out	<i>nummembers</i>	the number of elements that were read and thus the size of the returned array

Returns

an array of strings

Return values

<i>NULL</i>	something failed, errno might be set
-------------	--------------------------------------

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 585 of file kredis.c.

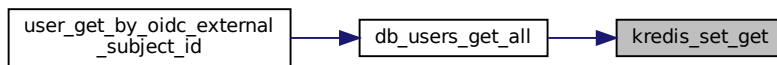
References [KREDIS_PREFIX](#), [kredis_set_free\(\)](#), and [kredis_s::rc](#).

Referenced by [db_users_get_all\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.2.15 kredis_set_ismember()

```
bool kredis_set_ismember (
    kredis_t * k,
    const char * key,
    const char * member )
```

Native Redis SSMEMBER Command.

Returns if member is a member of the set stored at key.

See also

<https://redis.io/commands/sismember>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>member</i>	the member

Returns

whether member is a member of the set

Return values

<i>true</i>	the element is a member of the set
<i>false</i>	the element is not a member of the set, key does not exist, or other failure

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 503 of file kredis.c.

References KREDIS_PREFIX, and kredis_s::rc.

5.13.2.16 kredis_set_num()

```
long long kredis_set_num (
    kredis_t * k,
    const char * key )
```

Native Redis SCARD Command.

Returns the set cardinality (number of elements) of the set stored at key.

See also

<https://redis.io/commands/scard>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key

Returns

the set cardinality

Return values

0	failed or there are no elements in the set
---	--

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 463 of file kredis.c.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

5.13.2.17 kredis_set_rem()

```
bool kredis_set_rem (
    kredis_t * k,
    const char * key,
    const char * member )
```

Native Redis SREM Command.

Remove the specified member from the set stored at key. Specified members that are not a member of this set are ignored. If key does not exist, it is treated as an empty set.

An error occurs when the value stored at key is not a set.

See also

<https://redis.io/commands/srem>

Parameters

<i>in, out</i>	<i>k</i>	the kredis context
<i>in</i>	<i>key</i>	the key
<i>in</i>	<i>member</i>	the member to be removed

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

Check for errors using `kredis_error()` after calling this function.

Definition at line 546 of file `kredis.c`.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

5.13.2.18 kredis_set_with_ex()

```
bool kredis_set_with_ex (
    kredis_t * k,
    const char * key,
    const char * value,
    long ex )
```

Native Redis SET Command with EX set.

Set key to hold the string value. If key already holds a value, it is overwritten, regardless of its type. Any previous time to live associated with the key is discarded on successful SET operation.

See also

<https://redis.io/commands/set>

Parameters

<i>in, out</i>	<i>k</i>	the kredis context
<i>in</i>	<i>key</i>	the key
<i>in</i>	<i>value</i>	the value
<i>in</i>	<i>ex</i>	expire time, in seconds

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

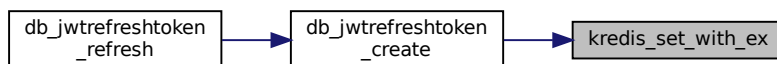
Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 333 of file kredis.c.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

Referenced by `db_jwtrefresh_token_create()`.

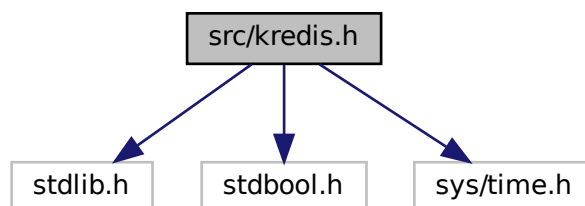
Here is the caller graph for this function:

**5.14 src/kredis.h File Reference**

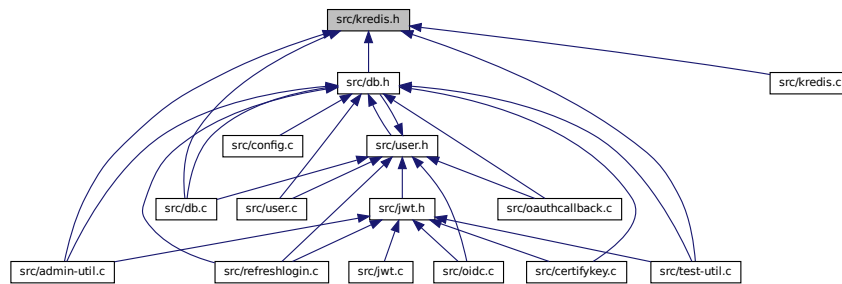
Header file for [kredis.c](#).

```
#include <stdlib.h>
#include <stdbool.h>
#include <sys/time.h>
```

Include dependency graph for kredis.h:



This graph shows which files directly or indirectly include this file:



Macros

- `#define KREDIS_PREFIX "sshca:"`
Extra prefix for every key.

Typedefs

- `typedef struct kredis_s kredis_t`
A kredis context.

Functions

- `kredis_t * kredis_connect` (const char *ip, int port, bool is_unix_socket, struct timeval timeout, const char *auth)
Connects to a redis server.
- `void kredis_free` (kredis_t *k)
Disconnects and frees a kredis_t object.
- `bool kredis_ping` (kredis_t *k)
Ping the redis server.
- `int kredis_error` (kredis_t *k)
Gets the current error code as an integer.
- `const char * kredis_error_string` (kredis_t *k)
Gets the current error code as a string.
- `char * kredis_get` (kredis_t *k, const char *key)
Native Redis GET Command.
- `bool kredis_set` (kredis_t *k, const char *key, const char *value)
Native Redis SET Command.
- `bool kredis_set_with_ex` (kredis_t *k, const char *key, const char *value, long ex)
Native Redis SET Command with EX set.
- `bool kredis_del` (kredis_t *k, const char *key)
Native Redis DEL Command.
- `bool kredis_set_add` (kredis_t *k, const char *key, const char *member)
Native Redis SADD Command.
- `long long kredis_set_num` (kredis_t *k, const char *key)
Native Redis SCARD Command.

- bool [kredis_set_ismember](#) ([kredis_t](#) *k, const char *key, const char *member)
Native Redis SISEMEMBER Command.
- char ** [kredis_set_get](#) ([kredis_t](#) *k, const char *key, size_t *nummembers)
Native Redis SMEMBERS Command.
- bool [kredis_set_rem](#) ([kredis_t](#) *k, const char *key, const char *member)
Native Redis SREM Command.
- void [kredis_set_free](#) (char **elements, size_t nummembers)
Frees the returned array from [kredis_set_get\(\)](#)
- bool [kredis_hash_set](#) ([kredis_t](#) *k, const char *key, const char *field, const char *value)
Native Redis HSET Command.
- bool [kredis_hash_exists](#) ([kredis_t](#) *k, const char *key, const char *field)
Native Redis HEXISTS Command.
- char * [kredis_hash_get](#) ([kredis_t](#) *k, const char *key, const char *field)
Native Redis HGET Command.

5.14.1 Detailed Description

Header file for [kredis.c](#).

5.14.2 Function Documentation

5.14.2.1 [kredis_connect\(\)](#)

```
kredis_t* kredis_connect (
    const char * ip,
    int port,
    bool is_unix_socket,
    struct timeval timeout,
    const char * auth )
```

Connects to a redis server.

Parameters

in	<i>ip</i>	The IP Address of the server or the path to a unix socket. If NULL set by default to 127.0.0.1.
in	<i>port</i>	The port of the redis server. Note: The default redis port is 6379.
in	<i>is_unix_socket</i>	Whether ip is a path to a unix socket. port is ignored if set.
in	<i>timeout</i>	Sets the connection timeout.
in	<i>auth</i>	The redis server password specified in the server config by requirepass. Ignored if equal to NULL.

Returns

On success a kredis context is returned. Otherwise NULL is returned and errno might be set.

The returned kredis context must be freed using [kredis_free\(\)](#).

Note

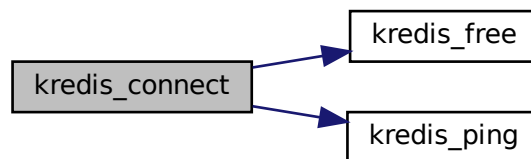
Only the old authentication method prior to Redis 6.0 is supported.

Definition at line 81 of file kredis.c.

References `kredis_free()`, `kredis_ping()`, and `kredis_s::rc`.

Referenced by `db_connect()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.2 `kredis_del()`

```
bool kredis_del (
    kredis_t * k,
    const char * key )
```

Native Redis DEL Command.

Removes the specified keys. A key is ignored if it does not exist.

See also

<https://redis.io/commands/del>

Parameters

<i>in, out</i>	<i>k</i>	the kredis context
<i>in</i>	<i>key</i>	the key

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 383 of file kredis.c.

References [KREDIS_PREFIX](#), and [kredis_s::rc](#).

Referenced by [db_jwtrefresh_token_refresh\(\)](#).

Here is the caller graph for this function:



5.14.2.3 kredis_error()

```
int kredis_error (
    kredis_t * k )
```

Gets the current error code as an integer.

In case an error actually happend, you can use [kredis_error_string\(\)](#) to get the error message as a string.

Parameters

<i>in</i>	<i>k</i>	the kredis context
-----------	----------	--------------------

Returns

the error code as an integer

Return values

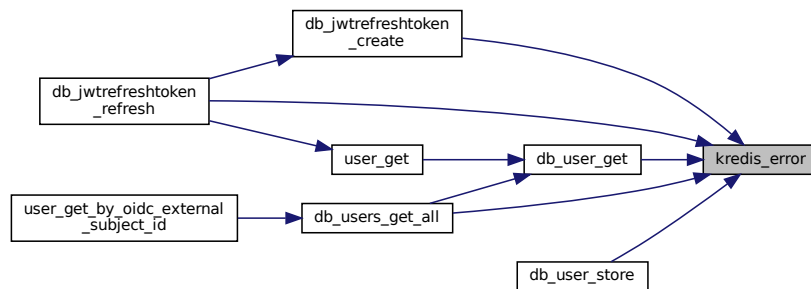
0	indicates no error
---	--------------------

Definition at line 192 of file kredis.c.

References kredis_s::rc.

Referenced by db_jwtrefresh_token_create(), db_jwtrefresh_token_refresh(), db_user_get(), db_user_store(), and db_users_get_all().

Here is the caller graph for this function:

**5.14.2.4 kredis_error_string()**

```
const char* kredis_error_string (
    kredis_t * k )
```

Gets the current error code as a string.

This function cannot be used for checking whether an error actually occurred, use [kredis_error\(\)](#) instead.

An empty string is returned if no error happend.

Parameters

in	k	the kredis context
----	---	--------------------

Returns

the error message

Return values

NULL	invalid argument or invalid redisContext
------	--

Note

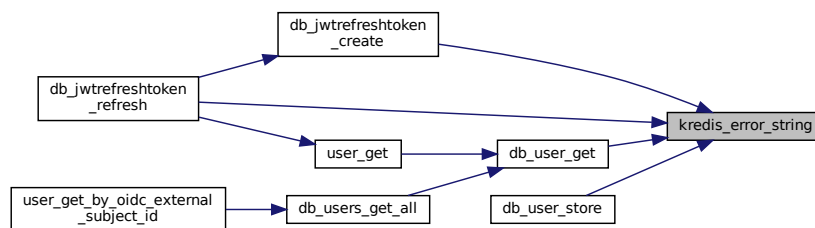
The returned string **MUST NOT** be freed.

Definition at line 213 of file kredis.c.

References kredis_s::rc.

Referenced by db_jwtrefresh_token_create(), db_jwtrefresh_token_refresh(), db_user_get(), and db_user_store().

Here is the caller graph for this function:



5.14.2.5 kredis_free()

```
void kredis_free (
    kredis_t * k )
```

Disconnects and frees a `kredis_t` object.

Parameters

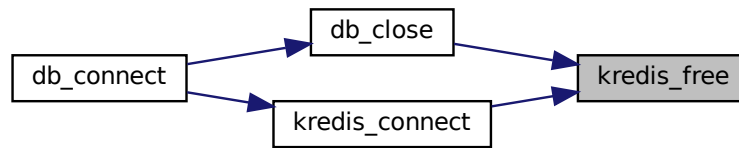
<code>in, out</code>	<code>k</code>	the kredis context
----------------------	----------------	--------------------

Definition at line 57 of file kredis.c.

References kredis_s::rc.

Referenced by `db_close()`, and `kredis_connect()`.

Here is the caller graph for this function:



5.14.2.6 kredis_get()

```
char* kredis_get (
    kredis_t * k,
    const char * key )
```

Native Redis GET Command.

Get the value of key.

The returned string must be freed using `free()`.

See also

<https://redis.io/commands/get>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key

Returns

the value of key

Return values

<i>NULL</i>	key doesn't exists or isn't a string
-------------	--------------------------------------

Note

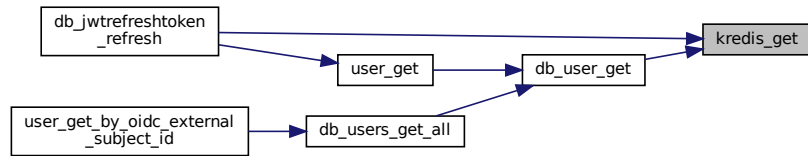
Check for errors using `kredis_error()` after calling this function.

Definition at line 238 of file `kredis.c`.

References KREDIS_PREFIX, and kredis_s::rc.

Referenced by db_jwtrefresh_token_refresh(), and db_user_get().

Here is the caller graph for this function:



5.14.2.7 kredis_hash_exists()

```

bool kredis_hash_exists (
    kredis_t * k,
    const char * key,
    const char * field )

```

Native Redis HEXISTS Command.

Returns if field is an existing field in the hash stored at key.

See also

<https://redis.io/commands/hexists>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>field</i>	the field

Returns

whether the field exists in the hash

Return values

<i>true</i>	the hash contains field
<i>false</i>	the hash does not contain field, key does not exist, or other failure

Note

Check for errors using `kredis_error()` after calling this function.

Definition at line 719 of file `kredis.c`.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

5.14.2.8 kredis_hash_get()

```
char* kredis_hash_get (
    kredis_t * k,
    const char * key,
    const char * field )
```

Native Redis HGET Command.

Returns the value associated with `field` in the hash stored at `key`.

The returned string must be freed using `free()`.

See also

<https://redis.io/commands/hget>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>field</i>	the field

Returns

the value

Return values

<code>NULL</code>	something failed, <code>errno</code> might be set
-------------------	---

Note

Check for errors using `kredis_error()` after calling this function.

Definition at line 759 of file `kredis.c`.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

5.14.2.9 kredis_hash_set()

```
bool kredis_hash_set (
    kredis_t * k,
    const char * key,
    const char * field,
    const char * value )
```

Native Redis HSET Command.

Sets field in the hash stored at key to value. If key does not exist, a new key holding a hash is created. If field already exists in the hash, it is overwritten.

See also

<https://redis.io/commands/hset>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>field</i>	the field
in	<i>value</i>	the value

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

Check for errors using `kredis_error()` after calling this function.

Definition at line 681 of file kredis.c.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

5.14.2.10 kredis_ping()

```
bool kredis_ping (
    kredis_t * k )
```

Ping the redis server.

Parameters

<i>in, out</i>	<i>k</i>	the kredis context
----------------	----------	--------------------

Returns

whether the ping was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

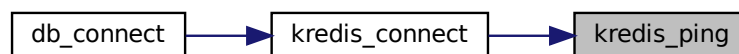
Check for errors using [kredis_error\(\)](#) on failure.

Definition at line 160 of file kredis.c.

References `kredis_s::rc`.

Referenced by `kredis_connect()`.

Here is the caller graph for this function:

**5.14.2.11 kredis_set()**

```

bool kredis_set (
    kredis_t * k,
    const char * key,
    const char * value )
  
```

Native Redis SET Command.

Set key to hold the string value. If key already holds a value, it is overwritten, regardless of its type. Any previous time to live associated with the key is discarded on successful SET operation.

See also

<https://redis.io/commands/set>

Parameters

<i>in, out</i>	<i>k</i>	the kredis context
<i>in</i>	<i>key</i>	the key
<i>in</i>	<i>value</i>	the value

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 284 of file kredis.c.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

Referenced by `db_user_store()`.

Here is the caller graph for this function:



5.14.2.12 kredis_set_add()

```

bool kredis_set_add (
    kredis_t * k,
    const char * key,
    const char * member )
  
```

Native Redis SADD Command.

Add the specified member to the set stored at key. Specified members that are already a member of this set are ignored. If key does not exist, a new set is created before adding the specified member.

An error occurs when the value stored at key is not a set.

See also

<https://redis.io/commands/sadd>

Parameters

<code>in, out</code>	<code>k</code>	the kredis context
<code>in</code>	<code>key</code>	the key
<code>in</code>	<code>member</code>	the member to be added

Returns

whether the command was successful.

Return values

<code>false</code>	failed
<code>true</code>	success

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 426 of file `kredis.c`.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

Referenced by `db_user_store()`.

Here is the caller graph for this function:

**5.14.2.13 kredis_set_free()**

```

void kredis_set_free (
    char ** elements,
    size_t nummembers )
  
```

Frees the returned array from [kredis_set_get\(\)](#)

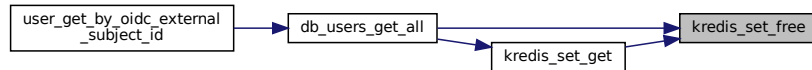
Parameters

<code>in, out</code>	<code>elements</code>	the array
<code>in</code>	<code>nummembers</code>	the size of the array

Definition at line 650 of file kredis.c.

Referenced by `db_users_get_all()`, and `kredis_set_get()`.

Here is the caller graph for this function:



5.14.2.14 kredis_set_get()

```
char** kredis_set_get (
    kredis_t * k,
    const char * key,
    size_t * nummembers )
```

Native Redis SMEMBERS Command.

Returns all the members of the set value stored at key.

The returned array must be freed using `kredis_set_free()`.

See also

<https://redis.io/commands/smembers>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
out	<i>nummembers</i>	the number of elements that were read and thus the size of the returned array

Returns

an array of strings

Return values

<i>NULL</i>	something failed, errno might be set
-------------	--------------------------------------

Note

Check for errors using `kredis_error()` after calling this function.

Definition at line 585 of file kredis.c.

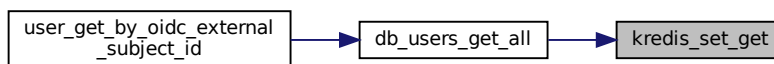
References KREDIS_PREFIX, kredis_set_free(), and kredis_s::rc.

Referenced by db_users_get_all().

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.2.15 kredis_set_ismember()

```

bool kredis_set_ismember (
    kredis_t * k,
    const char * key,
    const char * member )
  
```

Native Redis SISMEMBER Command.

Returns if member is a member of the set stored at key.

See also

<https://redis.io/commands/sismember>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>member</i>	the member

Returns

whether member is a member of the set

Return values

<i>true</i>	the element is a member of the set
<i>false</i>	the element is not a member of the set, key does not exist, or other failure

Note

Check for errors using `kredis_error()` after calling this function.

Definition at line 503 of file kredis.c.

References `KREDIS_PREFIX`, and `kredis_s::rc`.

5.14.2.16 kredis_set_num()

```
long long kredis_set_num (
    kredis_t * k,
    const char * key )
```

Native Redis SCARD Command.

Returns the set cardinality (number of elements) of the set stored at key.

See also

<https://redis.io/commands/scard>

Parameters

<i>in, out</i>	<i>k</i>	the kredis context
<i>in</i>	<i>key</i>	the key

Returns

the set cardinality

Return values

<i>0</i>	failed or there are no elements in the set
----------	--

Note

Check for errors using `kredis_error()` after calling this function.

Definition at line 463 of file kredis.c.

References KREDIS_PREFIX, and kredis_s::rc.

5.14.2.17 kredis_set_rem()

```
bool kredis_set_rem (
    kredis_t * k,
    const char * key,
    const char * member )
```

Native Redis SREM Command.

Remove the specified member from the set stored at key. Specified members that are not a member of this set are ignored. If key does not exist, it is treated as an empty set.

An error occurs when the value stored at key is not a set.

See also

<https://redis.io/commands/srem>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>member</i>	the member to be removed

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 546 of file kredis.c.

References KREDIS_PREFIX, and kredis_s::rc.

5.14.2.18 kredis_set_with_ex()

```
bool kredis_set_with_ex (
    kredis_t * k,
    const char * key,
    const char * value,
    long ex )
```

Native Redis SET Command with EX set.

Set key to hold the string value. If key already holds a value, it is overwritten, regardless of its type. Any previous time to live associated with the key is discarded on successful SET operation.

See also

<https://redis.io/commands/set>

Parameters

in, out	<i>k</i>	the kredis context
in	<i>key</i>	the key
in	<i>value</i>	the value
in	<i>ex</i>	expire time, in seconds

Returns

whether the command was successful.

Return values

<i>false</i>	failed
<i>true</i>	success

Note

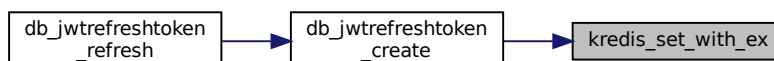
Check for errors using [kredis_error\(\)](#) after calling this function.

Definition at line 333 of file kredis.c.

References [KREDIS_PREFIX](#), and [kredis_s::rc](#).

Referenced by [db_jwtrefresh_token_create\(\)](#).

Here is the caller graph for this function:

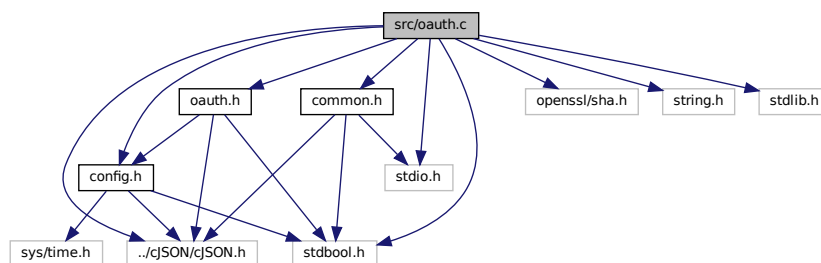


5.15 src/oauth.c File Reference

Useful code for OAuth Authentication.

```
#include "oauth.h"
#include "common.h"
#include "config.h"
#include "../cJSON/cJSON.h"
#include <openssl/sha.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
```

Include dependency graph for oauth.c:



Functions

- bool [oauth_verify_state_encoding](#) (const char *state)
Tests whether a state is encoded correctly.
- char * [oauth_gen_state](#) (void)
Generates a STATE for OAuth.
- char * [oauth_gen_code_verifier](#) (void)
Generates a CODE_VERIFIER for OAuth.
- char * [oauth_gen_code_challenge](#) (const char *code_verifier)
Generates a CODE_CHALLENGE for OAuth.
- char * [oauth_gen_url](#) (char *errmsg, const struct [config_auth_s](#) *authconfig, const char *state)
Generates an OAuth authorization Url.

5.15.1 Detailed Description

Useful code for OAuth Authentication.

5.15.2 Function Documentation

5.15.2.1 `oauth_gen_code_challenge()`

```
char* oauth_gen_code_challenge (
    const char * code_verifier )
```

Generates a `CODE_CHALLENGE` for OAuth.

This is a URL-safe base64-encoded (without padding) string of the SHA256 hash of the `CODE_VERIFIER`.

The returned string is already percent encoded.

See also

<https://docs.gitlab.com/ee/api/oauth2.html#authorization-code-with-proof-key-for-co>

Parameters

in	<code>code_verifier</code>	the <code>CODE_VERIFIER</code> needed for generating the <code>CODE_CHALLENGE</code>
----	----------------------------	--

Returns

the `CODE_CHALLENGE` represented by a string

Return values

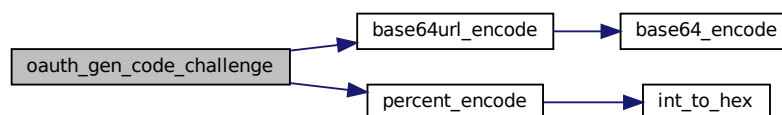
<code>NULL</code>	an error occurred and <code>errno</code> might be set
-------------------	---

The returned string must be freed using `free()`.

Definition at line 104 of file `oauth.c`.

References `base64url_encode()`, and `percent_encode()`.

Here is the call graph for this function:



5.15.2.2 `oauth_gen_code_verifier()`

```
char* oauth_gen_code_verifier (  
    void )
```

Generates a `CODE_VERIFIER` for OAuth.

This is basically just two consecutive UUIDv4s.

See also

<https://docs.gitlab.com/ee/api/oauth2.html#authorization-code-with-proof-key-for-co>

Returns

a new `CODE_VERIFIER` represented by a string

Return values

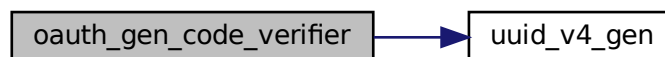
<i>NULL</i>	an error occurred and <code>errno</code> might be set
-------------	---

The returned string must be freed using `free()`.

Definition at line 70 of file `oauth.c`.

References `uuid_v4_gen()`, and `UUIDLEN`.

Here is the call graph for this function:



5.15.2.3 `oauth_gen_state()`

```
char* oauth_gen_state (  
    void )
```

Generates a `STATE` for OAuth.

This is basically just a UUIDv4.

See also

<https://docs.gitlab.com/ee/api/oauth2.html#authorization-code-with-proof-key-for-co>

Returns

a new `STATE` represented by a string

Return values

<code>NULL</code>	an error occurred and <code>errno</code> might be set
-------------------	---

The returned string must be freed using `free()`.

Definition at line 46 of file `oauth.c`.

References `uuid_v4_gen()`, and `UUIDLEN`.

Here is the call graph for this function:

5.15.2.4 `oauth_gen_url()`

```

char* oauth_gen_url (
    char * errmsg,
    const struct config_auth_s * authconfig,
    const char * state )
  
```

Generates an OAuth authorization Url.

An example `authconfig` can be found in `config.example.json["auth"]`.

See also

<https://docs.gitlab.com/ee/api/oauth2.html#authorization-code-flow>

Parameters

out	<code>errmsg</code>	buffer for error messages. size: <code>ERRMSGMAXSIZE</code>
in	<code>authconfig</code>	authentication settings
in	<code>state</code>	see oauth_gen_state()

Returns

A fully usable OAuth authorization Url.

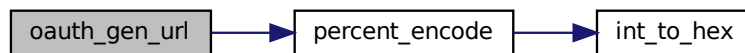
Return values

<i>NULL</i>	an error occurred. <code>errno</code> might be set and <code>errmsg</code> contains an error message.
-------------	---

Definition at line 134 of file `oauth.c`.

References `config_auth_s::app_id`, `config_auth_s::oauth_authorize_url`, `percent_encode()`, `config_auth_s::redirect_uri`, `config_auth_s::scope`, and `config_auth_s::type`.

Here is the call graph for this function:

**5.15.2.5 oauth_verify_state_encoding()**

```
bool oauth_verify_state_encoding (
    const char * state )
```

Tests whether a state is encoded correctly.

Parameters

in	<i>state</i>	see oauth_gen_state()
----	--------------	---------------------------------------

Returns

whether the state is encoded correctly

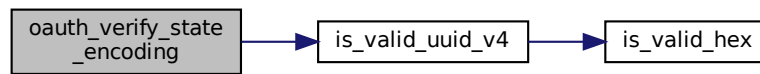
Return values

<i>true</i>	valid encoding; safe to use
<i>false</i>	invalid encoding; dangerous string!

Definition at line 30 of file `oauth.c`.

References `is_valid_uuid_v4()`.

Here is the call graph for this function:

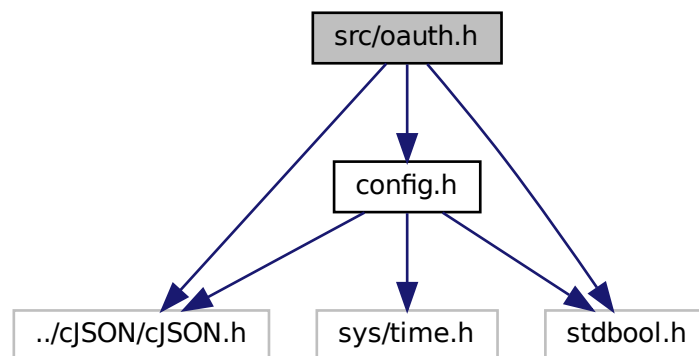


5.16 src/oauth.h File Reference

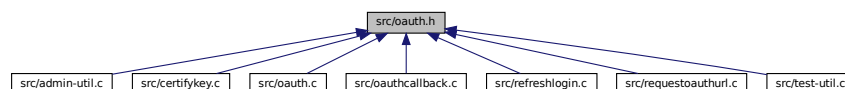
Header file for [oauth.c](#).

```
#include "config.h"
#include "../cJSON/cJSON.h"
#include <stdbool.h>
```

Include dependency graph for `oauth.h`:



This graph shows which files directly or indirectly include this file:



Functions

- bool [oauth_verify_state_encoding](#) (const char *state)

- Tests whether a state is encoded correctly.*
- char * `oauth_gen_state` (void)
Generates a STATE for OAuth.
 - char * `oauth_gen_code_verifier` (void)
Generates a CODE_VERIFIER for OAuth.
 - char * `oauth_gen_code_challenge` (const char *code_verifier)
Generates a CODE_CHALLENGE for OAuth.
 - char * `oauth_gen_url` (char *errmsg, const struct `config_auth_s` *authconfig, const char *state)
Generates an OAuth authorization Url.

5.16.1 Detailed Description

Header file for `oauth.c`.

5.16.2 Function Documentation

5.16.2.1 `oauth_gen_code_challenge()`

```
char* oauth_gen_code_challenge (
    const char * code_verifier )
```

Generates a CODE_CHALLENGE for OAuth.

This is a URL-safe base64-encoded (without padding) string of the SHA256 hash of the CODE_VERIFIER.

The returned string is already percent encoded.

See also

<https://docs.gitlab.com/ee/api/oauth2.html#authorization-code-with-proof-key-for-co>

Parameters

in	<code>code_verifier</code>	the CODE_VERIFIER needed for generating the CODE_CHALLENGE
----	----------------------------	--

Returns

the CODE_CHALLENGE represented by a string

Return values

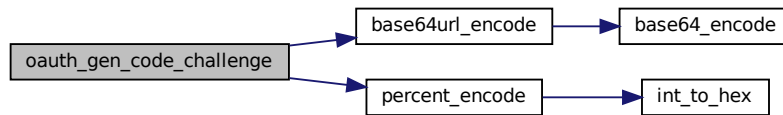
<code>NULL</code>	an error occurred and errno might be set
-------------------	--

The returned string must be freed using `free()`.

Definition at line 104 of file oauth.c.

References `base64url_encode()`, and `percent_encode()`.

Here is the call graph for this function:



5.16.2.2 `oauth_gen_code_verifier()`

```
char* oauth_gen_code_verifier (
    void )
```

Generates a `CODE_VERIFIER` for OAuth.

This is basically just two consecutive UUIDv4s.

See also

<https://docs.gitlab.com/ee/api/oauth2.html#authorization-code-with-proof-key-for-co>

Returns

a new `CODE_VERIFIER` represented by a string

Return values

<code>NULL</code>	an error occurred and <code>errno</code> might be set
-------------------	---

The returned string must be freed using `free()`.

Definition at line 70 of file oauth.c.

References `uuid_v4_gen()`, and `UUIDLEN`.

Here is the call graph for this function:



5.16.2.3 oauth_gen_state()

```
char* oauth_gen_state (
    void )
```

Generates a STATE for OAuth.

This is basically just a UUIDv4.

See also

<https://docs.gitlab.com/ee/api/oauth2.html#authorization-code-with-proof-key-for-co>

Returns

a new STATE represented by a string

Return values

<i>NULL</i>	an error occurred and errno might be set
-------------	--

The returned string must be freed using free().

Definition at line 46 of file oauth.c.

References `uuid_v4_gen()`, and `UUIDLEN`.

Here is the call graph for this function:



5.16.2.4 `oauth_gen_url()`

```
char* oauth_gen_url (
    char * errmsg,
    const struct config_auth_s * authconfig,
    const char * state )
```

Generates an OAuth authorization Url.

An example authconfig can be found in config.example.json["auth"].

See also

<https://docs.gitlab.com/ee/api/oauth2.html#authorization-code-flow>

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>authconfig</i>	authentication settings
in	<i>state</i>	see oauth_gen_state()

Returns

A fully usable OAuth authorization Url.

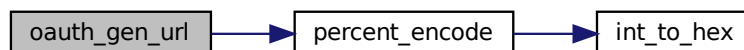
Return values

<i>NULL</i>	an error occurred. errno might be set and errmsg contains an error message.
-------------	---

Definition at line 134 of file oauth.c.

References `config_auth_s::app_id`, `config_auth_s::oauth_authorize_url`, `percent_encode()`, `config_auth_s::redirect_uri`, `config_auth_s::scope`, and `config_auth_s::type`.

Here is the call graph for this function:



5.16.2.5 `oauth_verify_state_encoding()`

```
bool oauth_verify_state_encoding (
    const char * state )
```

Tests whether a state is encoded correctly.

Parameters

in	<i>state</i>	see oauth_gen_state()
----	--------------	---------------------------------------

Returns

whether the state is encoded correctly

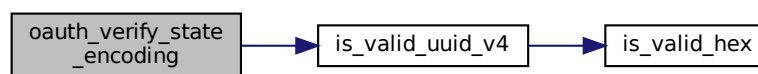
Return values

<i>true</i>	valid encoding; safe to use
<i>false</i>	invalid encoding; dangerous string!

Definition at line 30 of file `oauth.c`.

References `is_valid_uuid_v4()`.

Here is the call graph for this function:



5.17 src/oauthcallback.c File Reference

oauth callback receiver. handles endpoint at `/cgi/oauthcallback`

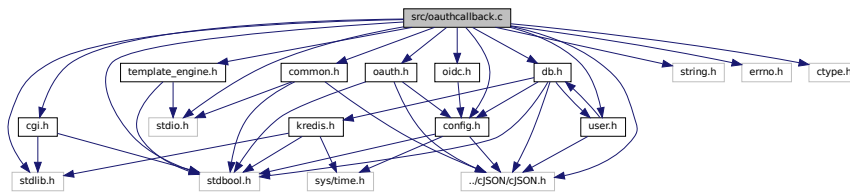
```

#include "common.h"
#include "cgi.h"
#include "db.h"
#include "oauth.h"
#include "config.h"
#include "user.h"
#include "oidc.h"
#include "template_engine.h"
#include "../cJSON/cJSON.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <errno.h>

```

```
#include <ctype.h>
```

Include dependency graph for oauthcallback.c:



Functions

- int [main](#) (void)
main function for oauthcallback

5.17.1 Detailed Description

oauth callback receiver. handles endpoint at /cgi/oauthcallback

Request: Expects a GET Request. The QUERY_STRING should contain:

- `code` the oauth code
- `state` the oauth state

Response: Content-Type: `text/html` The Body should contain the rendered version of `access_granted.html` or `access_denied.html`.

5.17.2 Function Documentation

5.17.2.1 main()

```
int main (
    void )
```

main function for oauthcallback

Returns

the exit code of this program.

Return values

0	success
!0	failure

Definition at line 44 of file oauthcallback.c.

References `cgi_testenv()`, `config_s::db`, `DB_JWTREFRESHTOKEN_BUFSIZE`, and `ERRMSGMAXSIZE`.

Here is the call graph for this function:



5.18 src/oidc.c File Reference

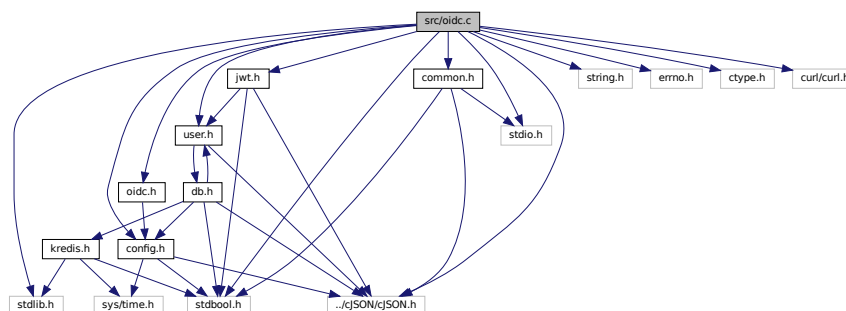
Open ID Connect implementation.

```

#include "oidc.h"
#include "user.h"
#include "common.h"
#include "config.h"
#include "jwt.h"
#include "../cJSON/cJSON.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
#include <curl/curl.h>

```

Include dependency graph for oidc.c:



Data Structures

- struct `oidc_memory_s`

internal struct used for receiving curl callback data

Functions

- char * [oidc_get_external_subject_id_from_identity_provider](#) (char *errmsg, const struct [config_auth_s](#) *authconfig, const char *code)

Get external subject id from identity provider.

5.18.1 Detailed Description

Open ID Connect implementation.

See also

[oauth.c](#)

[oauthcallback.c](#)

<https://goteleport.com/blog/how-oidc-authentication-works/>

5.18.2 Function Documentation

5.18.2.1 oidc_get_external_subject_id_from_identity_provider()

```
char* oidc_get_external_subject_id_from_identity_provider (
    char * errmsg,
    const struct config\_auth\_s * authconfig,
    const char * code )
```

Get external subject id from identity provider.

This function will use the oauth code received by the oauth callback endpoint to get the OpenID Connect id_token from the identity provider specified in the config.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>authconfig</i>	authentication settings
in	<i>code</i>	the oauth code oauthcallback received

Returns

the oidc external subject id of the user that just authenticated at the identity provider

Return values

<i>NULL</i>	an error occurred. errno might be set and errmsg contains an error message
-------------	--

The returned string must be freed using free().

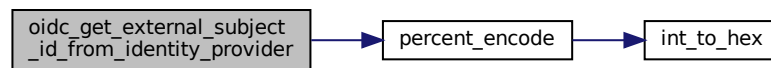
See also

[user_get_by_oidc_external_subject_id\(\)](#)

Definition at line 74 of file oidc.c.

References `config_auth_s::app_id`, `config_auth_s::app_secret`, `oidc_memory_s::memory`, `config_auth_s::oauth_token_url`, `percent_encode()`, `config_auth_s::redirect_uri`, `oidc_memory_s::size`, and `config_auth_s::type`.

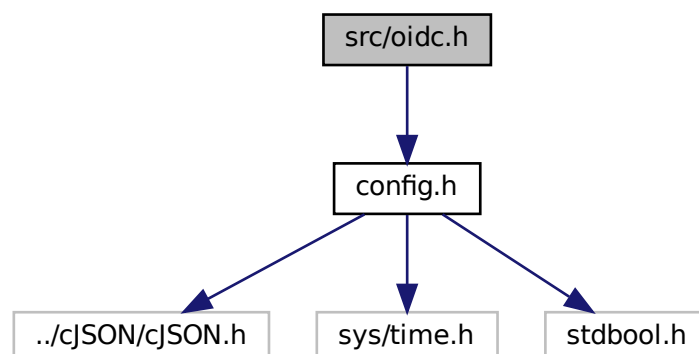
Here is the call graph for this function:



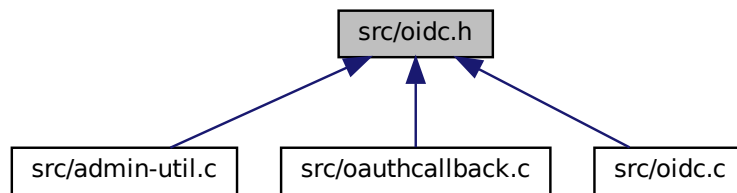
5.19 src/oidc.h File Reference

Header file for [oidc.c](#).

```
#include "config.h"  
Include dependency graph for oidc.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- char * [oidc_get_external_subject_id_from_identity_provider](#) (char *errmsg, const struct [config_auth_s](#) *authconfig, const char *code)

Get external subject id from identity provider.

5.19.1 Detailed Description

Header file for [oidc.c](#).

5.19.2 Function Documentation

5.19.2.1 oidc_get_external_subject_id_from_identity_provider()

```

char* oidc_get_external_subject_id_from_identity_provider (
    char * errmsg,
    const struct config\_auth\_s * authconfig,
    const char * code )
  
```

Get external subject id from identity provider.

This function will use the oauth code received by the oauth callback endpoint to get the OpenID Connect id_token from the identity provider specified in the config.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>authconfig</i>	authentication settings
in	<i>code</i>	the oauth code oautcallback received

Returns

the oidc external subject id of the user that just authenticated at the identity provider

Return values

<i>NULL</i>	an error occurred. errno might be set and errmsg contains an error message
-------------	--

The returned string must be freed using free().

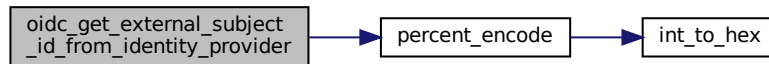
See also

[user_get_by_oidc_external_subject_id\(\)](#)

Definition at line 74 of file oidc.c.

References `config_auth_s::app_id`, `config_auth_s::app_secret`, `oidc_memory_s::memory`, `config_auth_s::oauth_url`, `percent_encode()`, `config_auth_s::redirect_uri`, `oidc_memory_s::size`, and `config_auth_s::type`.

Here is the call graph for this function:



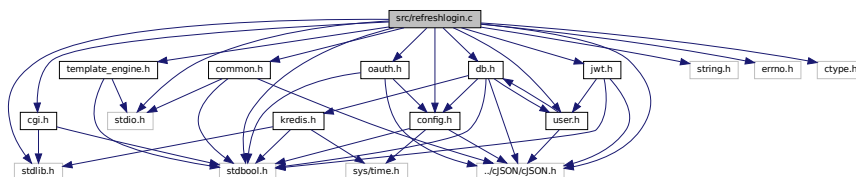
5.20 src/refreshlogin.c File Reference

utility for refreshing the jwtrefresh token and getting a JWT. handles endpoint at /cgi/refreshlogin

```

#include "common.h"
#include "cgi.h"
#include "db.h"
#include "jwt.h"
#include "oauth.h"
#include "config.h"
#include "user.h"
#include "template_engine.h"
#include "../cJSON/cJSON.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
  
```

Include dependency graph for refreshlogin.c:



Functions

- int `main` (void)

main function for refreshlogin

5.20.1 Detailed Description

utility for refreshing the jwtrefresh token and getting a JWT. handles endpoint at /cgi/refreshlogin

Request: Expects a POST Request with Content-Type set to `application/x-www-form-urlencoded`. The Request Header must contain a valid `jwtrefresh token` cookie. The User must be 'active'. The Body should be empty. The QUERY_STRING may contain a `as` field specifying the Content-Type of the response. (Supported are `json,html` and `plain`. Default: `json`)

Response as json (if successful): Content-Type: `application/json` Set-Cookie: `jwtrefresh token` The Body should contain a JSON Object with a `jwt` string field and an `expires_in` number field.

Response as html (if successful): Content-Type: `text/html` Set-Cookie: `jwtrefresh token` The Body should contain the rendered version of `refreshlogin.html`.

Response as plain (if successful): Content-Type: `text/plain` Set-Cookie: `jwtrefresh token` The Body should contain the JWT.

5.20.2 Function Documentation

5.20.2.1 `main()`

```
int main (
    void )
```

main function for refreshlogin

Returns

the exit code of this program.

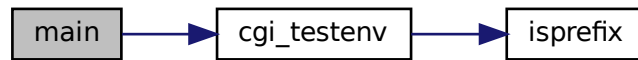
Return values

0	success
!0	failure

Definition at line 58 of file `refreshlogin.c`.

References `cgi_testenv()`, `config_s::db`, `DB_JWTREFRESH_TOKEN_BUFSIZE`, and `ERRMSGMAXSIZE`.

Here is the call graph for this function:



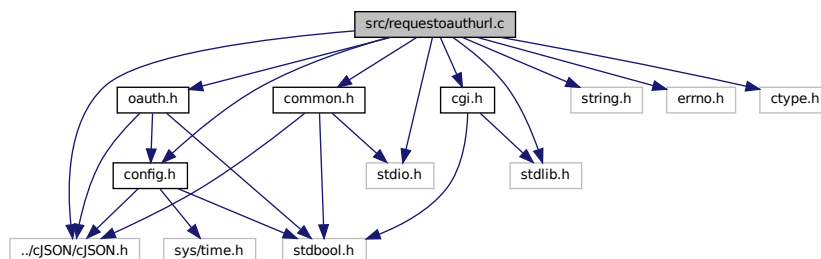
5.21 src/requestoathurl.c File Reference

oauth url generator. handles endpoint at /cgi/requestoathurl

```

#include "common.h"
#include "cgi.h"
#include "oauth.h"
#include "config.h"
#include "../cJSON/cJSON.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
  
```

Include dependency graph for requestoathurl.c:



Functions

- int [main](#) (void)

main function for requestoathurl

5.21.1 Detailed Description

oauth url generator. handles endpoint at /cgi/requestoathurl

Request: Expects a GET Request. The QUERY_STRING may contain a `as` field specifying the Content-Type of the response. (Supported are `json`, `redirect` and `plain`. Default: `json`)

Response as json (if successful): Content-Type: `application/json` The Body should contain a JSON Object with an `oathurl` string field.

Response as plain (if successful): Content-Type: `text/plain` The Body should contain `oathurl`.

Response as redirect (if successful): Content-Type: `text/html` Status: 302 Location: `oathurl`

5.21.2 Function Documentation

5.21.2.1 main()

```
int main (  
        void )
```

main function for requestoauthurl

Returns

the exit code of this program.

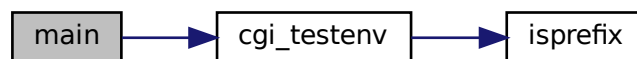
Return values

0	success
!0	failure

Definition at line 47 of file requestoauthurl.c.

References `cgi_testenv()`, and `ERRMSGMAXSIZE`.

Here is the call graph for this function:



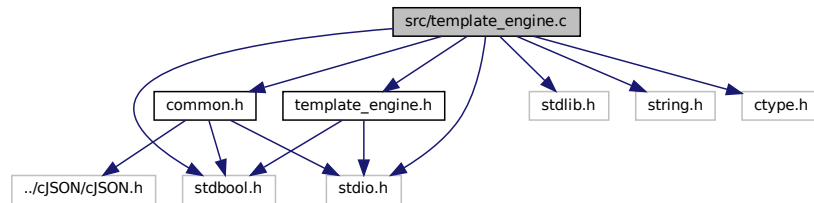
5.22 src/template_engine.c File Reference

Template rendering code.

```
#include "template_engine.h"  
#include "common.h"  
#include <stdbool.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>
```

```
#include <ctype.h>
```

Include dependency graph for `template_engine.c`:



Functions

- `char * tmplt_render` (`char *errmsg`, `const char *tmplt`, `const struct tmplt_kvps *data`)
Renders a template to a string.
- `char * tmplt_render_from_file` (`char *errmsg`, `const char *tmplt_file_path`, `const struct tmplt_kvps *data`)
Renders a template to a string.

5.22.1 Detailed Description

Template rendering code.

5.22.1.1 TEMPLATING LANGUAGE

The data model consists of key-value-pairs. A key in a template is marked by enclosing it in pairs of percent-signs. For example: "Hello %%name%%. I like you." Note: A template is invalid if a key doesn't exist.

5.22.2 Function Documentation

5.22.2.1 `tmplt_render()`

```
char* tmplt_render (
    char * errmsg,
    const char * tmplt,
    const struct tmplt\_kvps * data )
```

Renders a template to a string.

This will return a copy of `tmplt` where all instances of `%%KEY%%` are replaced with the associated value in `data`.

See also

[TEMPLATING LANGUAGE](#) // TODO reference section

A value marked as unsafe will be percent encoded using [percent_encode\(\)](#).

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>tmpl</i>	the template as a string
in	<i>data</i>	a null terminated key-value-pair array

Returns

the rendered document as a string

Return values

<i>NULL</i>	an error occurred and an error message has been written to errmsg. errno might be set.
-------------	--

The returned string must be freed using free().

Definition at line 66 of file template_engine.c.

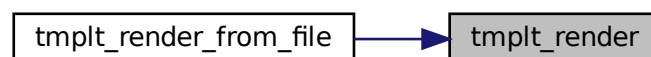
References `tmpl_kv_s::key`, `str_append()`, `TEMPLATE_KEY_CLOSE`, and `TEMPLATE_KEY_OPEN`.

Referenced by `tmpl_render_from_file()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.22.2.2 `tmpl_render_from_file()`

```
char* tmpl_render_from_file (
    char * errmsg,
    const char * tmpl_file_path,
    const struct tmpl\_kvps * data )
```

Renders a template to a string.

Wrapper for [tmpl_render\(\)](#).

See also

[tmpl_render\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>tmpl_file_path</i>	path to the template file
in	<i>data</i>	a null terminated key-value-pair array

Returns

the rendered document as a string

Return values

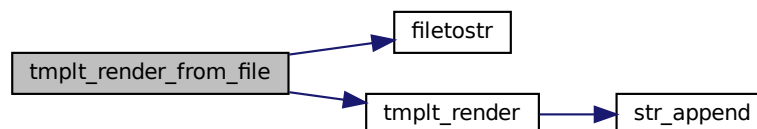
<i>NULL</i>	an error occurred and an error message has been written to <i>errmsg</i> . <i>errno</i> might be set.
-------------	---

The returned string must be freed using `free()`.

Definition at line 184 of file `template_engine.c`.

References `filetostr()`, and `tmpl_render()`.

Here is the call graph for this function:

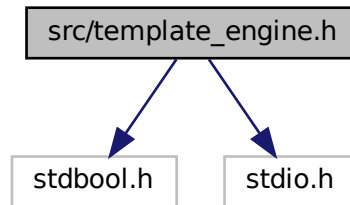


5.23 `src/template_engine.h` File Reference

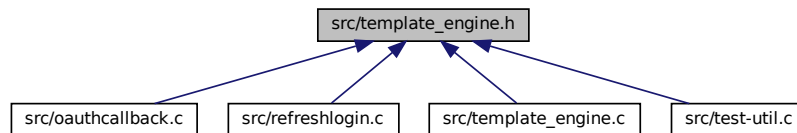
Header file for [template_engine.c](#).

```
#include <stdbool.h>
#include <stdio.h>
```

Include dependency graph for template_engine.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct [tmplt_kvp_s](#)
Template Key-Value-Pair Struct.

Macros

- #define [TEMPLATES_DIR](#) "../frontend/templates"
Path to the directory containing the templates.
- #define [TEMPLATE_KEY_OPEN](#) "%%"
Indicator for the beginning of a key.
- #define [TEMPLATE_KEY_CLOSE](#) "%%"
Indicator for the ending of a key.

Functions

- char * [tmplt_render](#) (char *errmsg, const char *tmplt, const struct [tmplt_kvp_s](#) *data)
Renders a template to a string.
- char * [tmplt_render_from_file](#) (char *errmsg, const char *tmplt_file_path, const struct [tmplt_kvp_s](#) *data)
Renders a template to a string.

5.23.1 Detailed Description

Header file for [template_engine.c](#).

5.23.2 Function Documentation

5.23.2.1 `tmpl_render()`

```
char* tmpl_render (
    char * errmsg,
    const char * tmpl,
    const struct tmpl\_kvps * data )
```

Renders a template to a string.

This will return a copy of `tmpl` where all instances of `%KEY%` are replaced with the associated value in `data`.

See also

TEMPLATING LANGUAGE // TODO reference section

A value marked as unsafe will be percent encoded using [percent_encode\(\)](#).

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>tmpl</i>	the template as a string
in	<i>data</i>	a null terminated key-value-pair array

Returns

the rendered document as a string

Return values

<i>NULL</i>	an error occurred and an error message has been written to <code>errmsg</code> . <code>errno</code> might be set.
-------------	---

The returned string must be freed using `free()`.

Definition at line 66 of file `template_engine.c`.

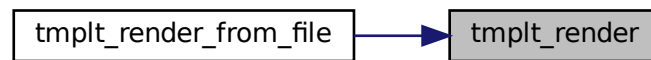
References `tmpl_kvps::key`, `str_append()`, `TEMPLATE_KEY_CLOSE`, and `TEMPLATE_KEY_OPEN`.

Referenced by `tmpl_render_from_file()`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.23.2.2 tmpplt_render_from_file()

```

char* tmpplt_render_from_file (
    char * errmsg,
    const char * tmpplt_file_path,
    const struct tmpplt_kvp_s * data )
  
```

Renders a template to a string.

Wrapper for [tmpplt_render\(\)](#).

See also

[tmpplt_render\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>tmpplt_file_path</i>	path to the template file
in	<i>data</i>	a null terminated key-value-pair array

Returns

the rendered document as a string

Return values

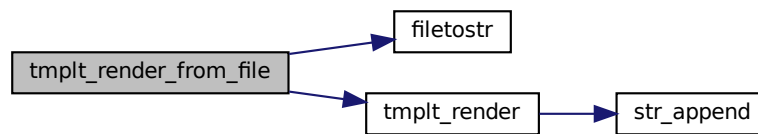
<code>NULL</code>	an error occurred and an error message has been written to <code>errmsg</code> . <code>errno</code> might be set.
-------------------	---

The returned string must be freed using `free()`.

Definition at line 184 of file `template_engine.c`.

References `filetostr()`, and `tmpl_render()`.

Here is the call graph for this function:



5.24 src/test-util.c File Reference

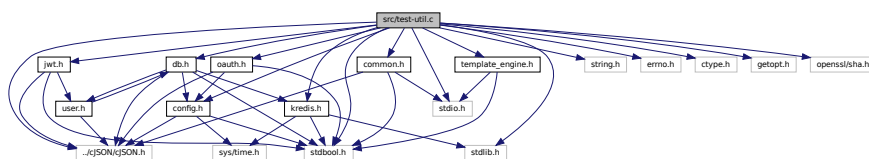
Tool for testing PARTs of this program.

```

#include "common.h"
#include "db.h"
#include "oauth.h"
#include "kredis.h"
#include "jwt.h"
#include "config.h"
#include "template_engine.h"
#include "../cJSON/cJSON.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include <errno.h>
#include <ctype.h>
#include <getopt.h>
#include <openssl/sha.h>

```

Include dependency graph for `test-util.c`:



Data Structures

- struct [testable_s](#)
testcase struct
- struct [encoder_testcase_s](#)
struct used for testing encoders

Macros

- #define [MAXFILEPATHLEN](#) 256
the maximum length of a file path

Functions

- int [main](#) (int argc, char **argv, char **envp)
main function for test-util

5.24.1 Detailed Description

Tool for testing PARTs of this program.

5.24.2 Function Documentation

5.24.2.1 main()

```
int main (  
    int argc,  
    char ** argv,  
    char ** envp )
```

main function for test-util

Parameters

in	<i>argc</i>	number of arguments from the command line
in	<i>argv</i>	arguments from the command line
in	<i>envp</i>	environment variables

Returns

the exit code of this program.

Return values

0	success
!0	failure

Definition at line 980 of file test-util.c.

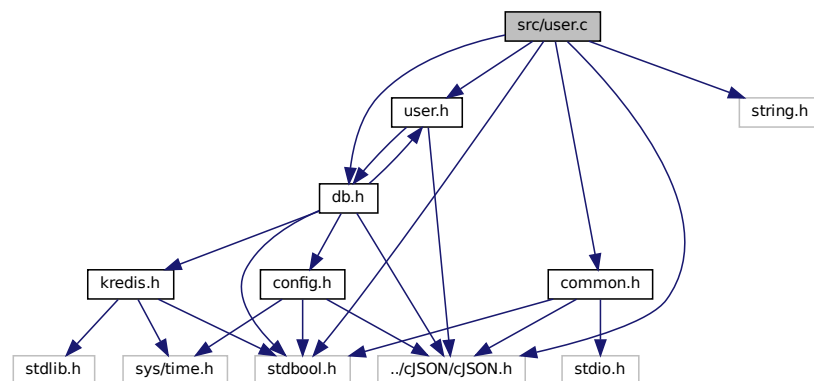
References ERRMSGMAXSIZE, and MAXFILEPATHLEN.

5.25 src/user.c File Reference

User related code.

```
#include "user.h"
#include "common.h"
#include "db.h"
#include "../cJSON/cJSON.h"
#include <string.h>
#include <stdbool.h>
```

Include dependency graph for user.c:



Functions

- [user_t * user_get_by_oidc_external_subject_id](#) (char *errmsg, [db_t](#) *db, const char *sub)
Retrieves a user from the database via their oidc external subject id.
- [user_t * user_get](#) (char *errmsg, [db_t](#) *db, const char *userid)
Retrieves a user from the database via their userid.
- [user_t * user_parse](#) (char *errmsg, const cJSON *userjson)
Parses a user from json.
- void [user_free](#) ([user_t](#) *user)
Frees a user.

5.25.1 Detailed Description

User related code.

5.25.2 Function Documentation

5.25.2.1 user_free()

```
void user_free (  
    user_t * user )
```

Frees a user.

Parameters

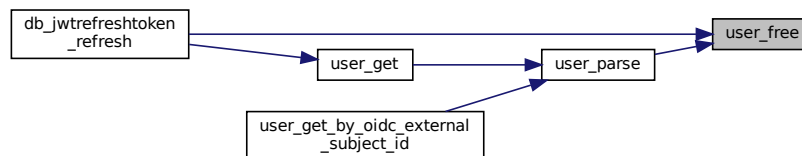
in	<i>user</i>	the user
----	-------------	----------

Definition at line 212 of file user.c.

References `user_s::__raw`, and `user_s::principals`.

Referenced by `db_jwtrefresh_token_refresh()`, and `user_parse()`.

Here is the caller graph for this function:



5.25.2.2 user_get()

```
user_t* user_get (  
    char * errmsg,  
    db_t * db,  
    const char * userid )
```

Retrieves a user from the database via their userid.

See also

[db_user_get\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
in	<i>userid</i>	the users userid

Returns

the user

Return values

<i>NULL</i>	an error occurred or user doesn't exist in the database. An error message has been written to errmsg.
-------------	---

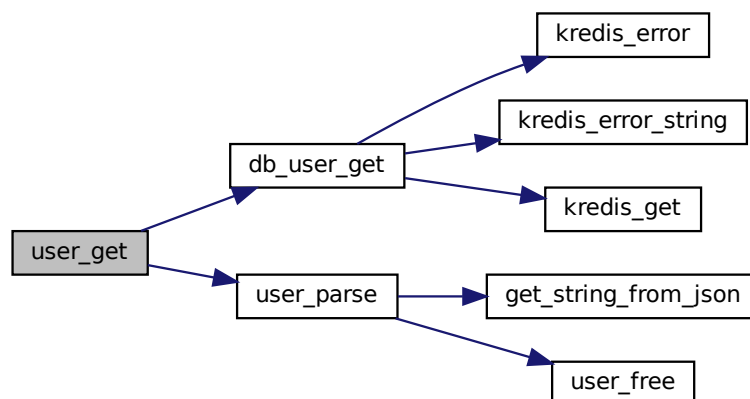
The returned user object must be freed using [user_free\(\)](#).

Definition at line 82 of file user.c.

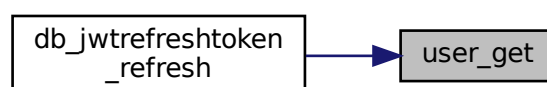
References [db_user_get\(\)](#), [testable_s::errmsg](#), [testable_s::res](#), and [user_parse\(\)](#).

Referenced by [db_jwtrefresh_token_refresh\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.25.2.3 user_get_by_oidc_external_subject_id()

```
user_t* user_get_by_oidc_external_subject_id (
    char * errmsg,
    db_t * db,
    const char * sub )
```

Retrieves a user from the database via their oidc external subject id.

See also

[oidc_get_external_subject_id_from_identity_provider\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
in	<i>sub</i>	the oidc external subject id

Returns

the user

Return values

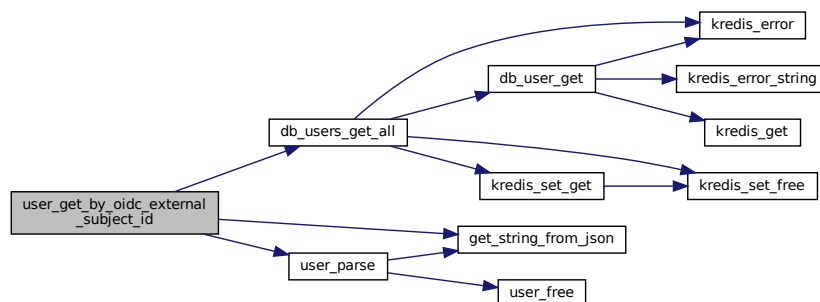
<i>NULL</i>	an error occurred or user doesn't exist in the database. An error message has been written to errmsg.
-------------	---

The returned user object must be freed using [user_free\(\)](#).

Definition at line 31 of file user.c.

References [db_users_get_all\(\)](#), [testable_s::errmsg](#), [get_string_from_json\(\)](#), [testable_s::res](#), and [user_parse\(\)](#).

Here is the call graph for this function:



5.25.2.4 user_parse()

```
user_t* user_parse (
    char * errmsg,
    const cJSON * userjson )
```

Parses a user from json.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>userjson</i>	the user as json

Returns

the parsed user

Return values

<i>NULL</i>	an error occurred and an error message has been written to errmsg
-------------	---

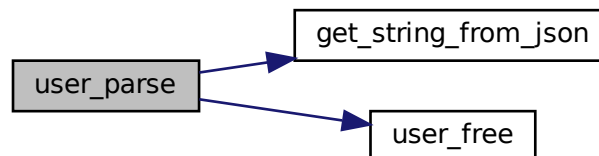
The returned user object must be freed using [user_free\(\)](#).

Definition at line 107 of file user.c.

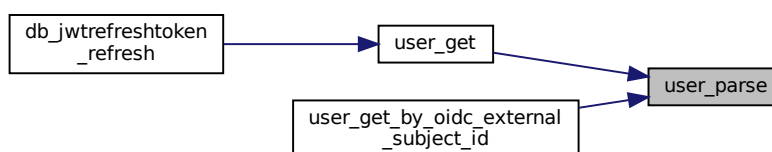
References `testable_s::errmsg`, `get_string_from_json()`, `testable_s::res`, and `user_free()`.

Referenced by `user_get()`, and `user_get_by_oidc_external_subject_id()`.

Here is the call graph for this function:



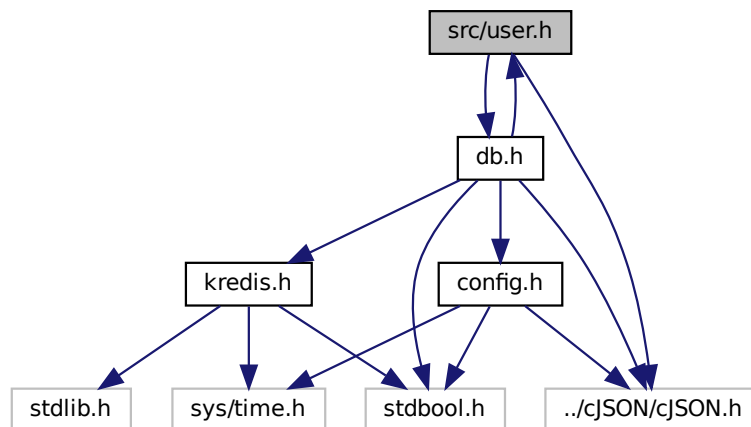
Here is the caller graph for this function:



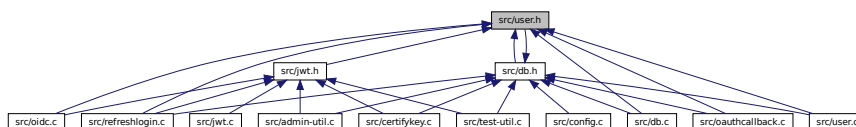
5.26 src/user.h File Reference

Header file for [user.c](#).

```
#include "db.h"
#include "../cJSON/cJSON.h"
Include dependency graph for user.h:
```



This graph shows which files directly or indirectly include this file:



Data Structures

- struct `user_s`
user struct

Typedefs

- typedef struct `user_s` `user_t`

Functions

- `user_t * user_get_by_oidc_external_subject_id` (char *errmsg, db_t *db, const char *sub)
Retrieves a user from the database via their oidc external subject id.
- `user_t * user_get` (char *errmsg, db_t *db, const char *userid)
Retrieves a user from the database via their userid.
- `user_t * user_parse` (char *errmsg, const cJSON *userjson)
Parses a user from json.
- void `user_free` (user_t *user)
Frees a user.

5.26.1 Detailed Description

Header file for `user.c`.

5.26.2 Function Documentation

5.26.2.1 user_free()

```
void user_free (
    user_t * user )
```

Frees a user.

Parameters

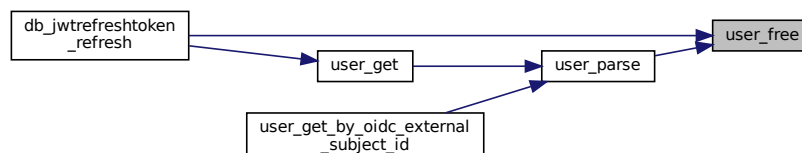
in	<code>user</code>	the user
----	-------------------	----------

Definition at line 212 of file `user.c`.

References `user_s::__raw`, and `user_s::principals`.

Referenced by `db_jwtrefresh_token_refresh()`, and `user_parse()`.

Here is the caller graph for this function:



5.26.2.2 user_get()

```
user_t* user_get (
    char * errmsg,
    db_t * db,
    const char * userid )
```

Retrieves a user from the database via their userid.

See also

[db_user_get\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
in	<i>userid</i>	the users userid

Returns

the user

Return values

<i>NULL</i>	an error occurred or user doesn't exist in the database. An error message has been written to errmsg.
-------------	---

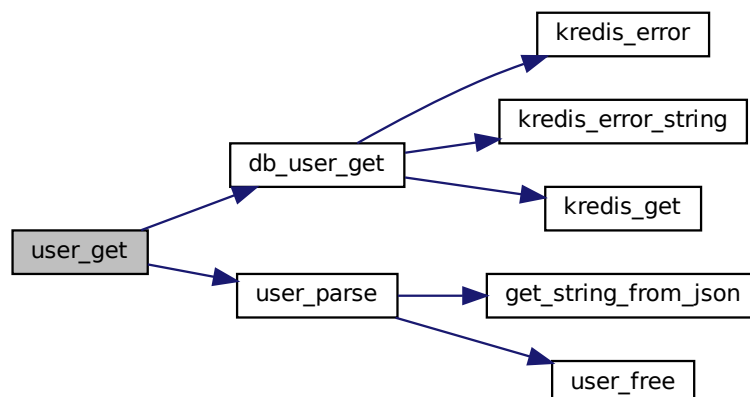
The returned user object must be freed using [user_free\(\)](#).

Definition at line 82 of file user.c.

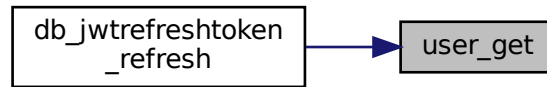
References [db_user_get\(\)](#), `testable_s::errmsg`, `testable_s::res`, and [user_parse\(\)](#).

Referenced by [db_jwtrefresh_token_refresh\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.26.2.3 user_get_by_oidc_external_subject_id()

```

user_t* user_get_by_oidc_external_subject_id (
    char * errmsg,
    db_t * db,
    const char * sub )
  
```

Retrieves a user from the database via their oidc external subject id.

See also

[oidc_get_external_subject_id_from_identity_provider\(\)](#)

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in, out	<i>db</i>	the database context
in	<i>sub</i>	the oidc external subject id

Returns

the user

Return values

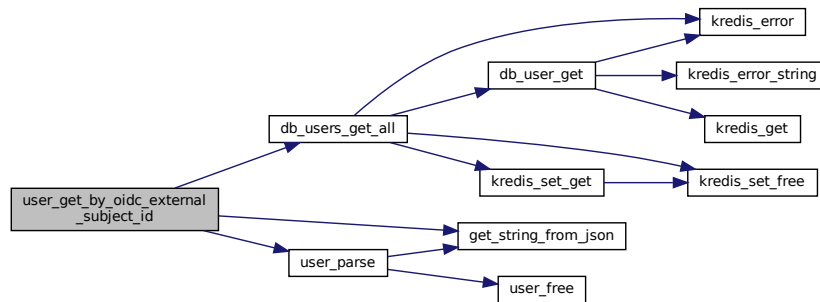
<i>NULL</i>	an error occurred or user doesn't exist in the database. An error message has been written to errmsg.
-------------	---

The returned user object must be freed using [user_free\(\)](#).

Definition at line 31 of file user.c.

References [db_users_get_all\(\)](#), [testable_s::errmsg](#), [get_string_from_json\(\)](#), [testable_s::res](#), and [user_parse\(\)](#).

Here is the call graph for this function:



5.26.2.4 user_parse()

```

user_t* user_parse (
    char * errmsg,
    const cJSON * userjson )
  
```

Parses a user from json.

Parameters

out	<i>errmsg</i>	buffer for error messages. size: ERRMSGMAXSIZE
in	<i>userjson</i>	the user as json

Returns

the parsed user

Return values

<i>NULL</i>	an error occurred and an error message has been written to errmsg
-------------	---

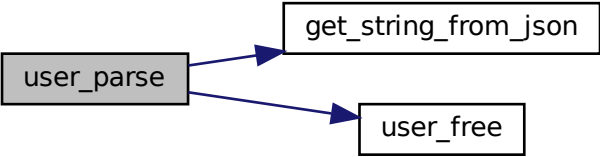
The returned user object must be freed using [user_free\(\)](#).

Definition at line 107 of file user.c.

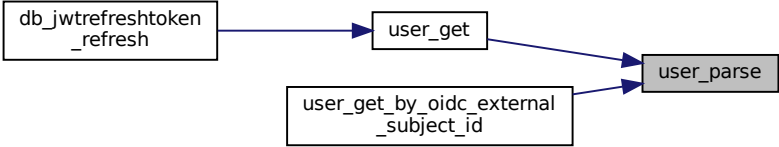
References [testable_s::errmsg](#), [get_string_from_json\(\)](#), [testable_s::res](#), and [user_free\(\)](#).

Referenced by [user_get\(\)](#), and [user_get_by_oidc_external_subject_id\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



Index

- admin-util.c
 - main, [24](#)
- base64_decode
 - common.c, [48](#)
 - common.h, [69](#)
- base64_encode
 - common.c, [48](#)
 - common.h, [69](#)
- base64url_decode
 - common.c, [49](#)
 - common.h, [70](#)
- base64url_encode
 - common.c, [50](#)
 - common.h, [71](#)
- certifykey.c
 - main, [26](#)
- cgi.c
 - cgi_kv_p_add, [27](#)
 - cgi_kv_p_add_delimbased, [29](#)
 - cgi_kv_p_create, [30](#)
 - cgi_kv_p_free, [31](#)
 - cgi_kv_p_get, [32](#)
 - cgi_parse_http_cookie, [33](#)
 - cgi_parse_query_string, [34](#)
 - cgi_testenv, [35](#)
- cgi.h
 - cgi_kv_p_add, [37](#)
 - cgi_kv_p_add_delimbased, [38](#)
 - cgi_kv_p_create, [40](#)
 - cgi_kv_p_free, [40](#)
 - cgi_kv_p_get, [41](#)
 - cgi_parse_http_cookie, [43](#)
 - cgi_parse_query_string, [44](#)
 - cgi_testenv, [45](#)
- cgi_kv_p_add
 - cgi.c, [27](#)
 - cgi.h, [37](#)
- cgi_kv_p_add_delimbased
 - cgi.c, [29](#)
 - cgi.h, [38](#)
- cgi_kv_p_create
 - cgi.c, [30](#)
 - cgi.h, [40](#)
- cgi_kv_p_free
 - cgi.c, [31](#)
 - cgi.h, [40](#)
- cgi_kv_p_get
 - cgi.c, [32](#)
 - cgi.h, [41](#)
- cgi_kv_s, [7](#)
- cgi_parse_http_cookie
 - cgi.c, [33](#)
 - cgi.h, [43](#)
- cgi_parse_query_string
 - cgi.c, [34](#)
 - cgi.h, [44](#)
- cgi_testenv
 - cgi.c, [35](#)
 - cgi.h, [45](#)
- command_arg_s, [8](#)
- command_s, [9](#)
- common.c
 - base64_decode, [48](#)
 - base64_encode, [48](#)
 - base64url_decode, [49](#)
 - base64url_encode, [50](#)
 - filetojson, [51](#)
 - filetostr, [52](#)
 - get_bool_from_json, [53](#)
 - get_number_from_json, [54](#)
 - get_string_from_json, [55](#)
 - hex_to_int, [55](#)
 - hmac_sha256, [56](#)
 - int_to_hex, [57](#)
 - is_percent_encoded, [58](#)
 - is_valid_hex, [58](#)
 - is_valid_uuid_v4, [59](#)
 - isprefix, [60](#)
 - percent_decode, [61](#)
 - percent_encode, [62](#)
 - remove_whitespace, [63](#)
 - str_append, [64](#)
 - uuid_v4_gen, [65](#)
 - writestreamtofile, [66](#)
- common.h
 - base64_decode, [69](#)
 - base64_encode, [69](#)
 - base64url_decode, [70](#)
 - base64url_encode, [71](#)
 - filetojson, [72](#)
 - filetostr, [73](#)
 - get_bool_from_json, [74](#)
 - get_number_from_json, [75](#)
 - get_string_from_json, [76](#)
 - hex_to_int, [76](#)
 - hmac_sha256, [77](#)
 - int_to_hex, [78](#)

- is_percent_encoded, 79
- is_valid_hex, 79
- is_valid_uuid_v4, 80
- isprefix, 81
- percent_decode, 82
- percent_encode, 83
- remove_whitespace, 84
- str_append, 85
- uuid_v4_gen, 86
- writestreamtofile, 87
- config.c
 - config_free, 89
 - config_parse, 89
- config.h
 - config_free, 91
 - config_parse, 91
- config_auth_s, 10
- config_certify_s, 12
- config_db_s, 13
- config_free
 - config.c, 89
 - config.h, 91
- config_jwt_s, 14
- config_parse
 - config.c, 89
 - config.h, 91
- config_s, 15
- db.c
 - db_close, 94
 - db_connect, 94
 - db_jwtrefresh_token_create, 95
 - db_jwtrefresh_token_refresh, 97
 - db_user_get, 98
 - db_user_store, 99
 - db_users_get_all, 101
- db.h
 - db_close, 104
 - db_connect, 105
 - db_jwtrefresh_token_create, 106
 - db_jwtrefresh_token_refresh, 108
 - DB_PREFIX_USERS, 104
 - db_user_get, 109
 - db_user_store, 110
 - db_users_get_all, 112
- db_close
 - db.c, 94
 - db.h, 104
- db_connect
 - db.c, 94
 - db.h, 105
- db_jwtrefresh_token_create
 - db.c, 95
 - db.h, 106
- db_jwtrefresh_token_refresh
 - db.c, 97
 - db.h, 108
- DB_PREFIX_USERS
 - db.h, 104
- db_s, 16
- db_user_get
 - db.c, 98
 - db.h, 109
- db_user_store
 - db.c, 99
 - db.h, 110
- db_users_get_all
 - db.c, 101
 - db.h, 112
- encoder_testcase_s, 17
- filetojson
 - common.c, 51
 - common.h, 72
- filetostr
 - common.c, 52
 - common.h, 73
- get_bool_from_json
 - common.c, 53
 - common.h, 74
- get_number_from_json
 - common.c, 54
 - common.h, 75
- get_string_from_json
 - common.c, 55
 - common.h, 76
- hex_to_int
 - common.c, 55
 - common.h, 76
- hmac_sha256
 - common.c, 56
 - common.h, 77
- int_to_hex
 - common.c, 57
 - common.h, 78
- is_percent_encoded
 - common.c, 58
 - common.h, 79
- is_valid_hex
 - common.c, 58
 - common.h, 79
- is_valid_uuid_v4
 - common.c, 59
 - common.h, 80
- isprefix
 - common.c, 60
 - common.h, 81
- jwt.c
 - jwt_decode, 114
 - jwt_encode_and_sign, 115
 - jwt_gen_payload, 117
 - jwt_has_expired, 118
 - jwt_simple_decode, 119
 - jwt_simple_encode, 120

- jwt_verify_encoding, 121
- jwt_verify_signature, 122
- jwt.h
 - jwt_decode, 124
 - jwt_encode_and_sign, 125
 - jwt_gen_payload, 127
 - jwt_has_expired, 128
 - jwt_simple_decode, 129
 - jwt_simple_encode, 130
 - jwt_verify_encoding, 131
 - jwt_verify_signature, 132
- jwt_decode
 - jwt.c, 114
 - jwt.h, 124
- jwt_encode_and_sign
 - jwt.c, 115
 - jwt.h, 125
- jwt_gen_payload
 - jwt.c, 117
 - jwt.h, 127
- jwt_has_expired
 - jwt.c, 118
 - jwt.h, 128
- jwt_simple_decode
 - jwt.c, 119
 - jwt.h, 129
- jwt_simple_encode
 - jwt.c, 120
 - jwt.h, 130
- jwt_verify_encoding
 - jwt.c, 121
 - jwt.h, 131
- jwt_verify_signature
 - jwt.c, 122
 - jwt.h, 132
- kredis.c
 - kredis_connect, 135
 - kredis_del, 136
 - kredis_error, 137
 - kredis_error_string, 138
 - kredis_free, 139
 - kredis_get, 139
 - kredis_hash_exists, 140
 - kredis_hash_get, 141
 - kredis_hash_set, 142
 - kredis_ping, 143
 - kredis_set, 144
 - kredis_set_add, 145
 - kredis_set_free, 146
 - kredis_set_get, 146
 - kredis_set_ismember, 148
 - kredis_set_num, 148
 - kredis_set_rem, 149
 - kredis_set_with_ex, 150
- kredis.h
 - kredis_connect, 153
 - kredis_del, 154
 - kredis_error, 155
 - kredis_error_string, 156
 - kredis_free, 157
 - kredis_get, 158
 - kredis_hash_exists, 159
 - kredis_hash_get, 160
 - kredis_hash_set, 160
 - kredis_ping, 161
 - kredis_set, 162
 - kredis_set_add, 163
 - kredis_set_free, 164
 - kredis_set_get, 165
 - kredis_set_ismember, 166
 - kredis_set_num, 167
 - kredis_set_rem, 168
 - kredis_set_with_ex, 168
- kredis_connect
 - kredis.c, 135
 - kredis.h, 153
- kredis_del
 - kredis.c, 136
 - kredis.h, 154
- kredis_error
 - kredis.c, 137
 - kredis.h, 155
- kredis_error_string
 - kredis.c, 138
 - kredis.h, 156
- kredis_free
 - kredis.c, 139
 - kredis.h, 157
- kredis_get
 - kredis.c, 139
 - kredis.h, 158
- kredis_hash_exists
 - kredis.c, 140
 - kredis.h, 159
- kredis_hash_get
 - kredis.c, 141
 - kredis.h, 160
- kredis_hash_set
 - kredis.c, 142
 - kredis.h, 160
- kredis_ping
 - kredis.c, 143
 - kredis.h, 161
- kredis_s, 18
- kredis_set
 - kredis.c, 144
 - kredis.h, 162
- kredis_set_add
 - kredis.c, 145
 - kredis.h, 163
- kredis_set_free
 - kredis.c, 146
 - kredis.h, 164
- kredis_set_get
 - kredis.c, 146
 - kredis.h, 165

- kredis_set_ismember
 - kredis.c, 148
 - kredis.h, 166
- kredis_set_num
 - kredis.c, 148
 - kredis.h, 167
- kredis_set_rem
 - kredis.c, 149
 - kredis.h, 168
- kredis_set_with_ex
 - kredis.c, 150
 - kredis.h, 168
- main
 - admin-util.c, 24
 - certifykey.c, 26
 - oauthcallback.c, 181
 - refreshlogin.c, 187
 - requestoauthurl.c, 189
 - test-util.c, 197
- oauth.c
 - oauth_gen_code_challenge, 170
 - oauth_gen_code_verifier, 171
 - oauth_gen_state, 172
 - oauth_gen_url, 173
 - oauth_verify_state_encoding, 174
- oauth.h
 - oauth_gen_code_challenge, 176
 - oauth_gen_code_verifier, 177
 - oauth_gen_state, 178
 - oauth_gen_url, 178
 - oauth_verify_state_encoding, 179
- oauth_gen_code_challenge
 - oauth.c, 170
 - oauth.h, 176
- oauth_gen_code_verifier
 - oauth.c, 171
 - oauth.h, 177
- oauth_gen_state
 - oauth.c, 172
 - oauth.h, 178
- oauth_gen_url
 - oauth.c, 173
 - oauth.h, 178
- oauth_verify_state_encoding
 - oauth.c, 174
 - oauth.h, 179
- oauthcallback.c
 - main, 181
- oidc.c
 - oidc_get_external_subject_id_from_identity_provider, 183
- oidc.h
 - oidc_get_external_subject_id_from_identity_provider, 185
- oidc_get_external_subject_id_from_identity_provider
 - oidc.c, 183
 - oidc.h, 185
- oidc_memory_s, 19
- percent_decode
 - common.c, 61
 - common.h, 82
- percent_encode
 - common.c, 62
 - common.h, 83
- refreshlogin.c
 - main, 187
- remove_whitespace
 - common.c, 63
 - common.h, 84
- requestoauthurl.c
 - main, 189
- src/admin-util.c, 23
- src/certifykey.c, 25
- src/cgi.c, 26
- src/cgi.h, 36
- src/common.c, 46
- src/common.h, 66
- src/config.c, 87
- src/config.h, 90
- src/db.c, 92
- src/db.h, 102
- src/jwt.c, 113
- src/jwt.h, 123
- src/kredis.c, 133
- src/kredis.h, 151
- src/oauth.c, 170
- src/oauth.h, 175
- src/oauthcallback.c, 180
- src/oidc.c, 182
- src/oidc.h, 184
- src/refreshlogin.c, 186
- src/requestoauthurl.c, 188
- src/template_engine.c, 189
- src/template_engine.h, 192
- src/test-util.c, 196
- src/user.c, 198
- src/user.h, 203
- str_append
 - common.c, 64
 - common.h, 85
- template_engine.c
 - tmpl_render, 190
 - tmpl_render_from_file, 191
- template_engine.h
 - tmpl_render, 194
 - tmpl_render_from_file, 195
- test-util.c
 - main, 197
- testable_s, 20
- tmpl_kv_s, 21
- tmpl_render
 - template_engine.c, 190

- template_engine.h, [194](#)
- tmpl_render_from_file
 - template_engine.c, [191](#)
 - template_engine.h, [195](#)
- user.c
 - user_free, [199](#)
 - user_get, [199](#)
 - user_get_by_oidc_external_subject_id, [201](#)
 - user_parse, [201](#)
- user.h
 - user_free, [204](#)
 - user_get, [204](#)
 - user_get_by_oidc_external_subject_id, [206](#)
 - user_parse, [207](#)
- user_free
 - user.c, [199](#)
 - user.h, [204](#)
- user_get
 - user.c, [199](#)
 - user.h, [204](#)
- user_get_by_oidc_external_subject_id
 - user.c, [201](#)
 - user.h, [206](#)
- user_parse
 - user.c, [201](#)
 - user.h, [207](#)
- user_s, [22](#)
- uuid_v4_gen
 - common.c, [65](#)
 - common.h, [86](#)
- writestreamtofile
 - common.c, [66](#)
 - common.h, [87](#)