NAME

oauth.h -

OAuth.net implementation in POSIX-C.

SYNOPSIS

Enumerations

enum **OAuthMethod** { **OA_HMAC** =0, **OA_RSA**, **OA_PLAINTEXT** }
signature method to used for signing the request.

Functions

char * **oauth_encode_base64** (int size, const unsigned char *src)
Base64 encode and return size data in 'src'.
int **oauth_decode_base64** (unsigned char *dest, const char *src)
Decode the base64 encoded string 'src' into the memory pointed to by 'dest'.
char * **oauth_url_escape** (const char *string)
Escape 'string' according to RFC3986 and http://oauth.net/core/1.0/#encoding_parameters.
char * **oauth_url_unescape** (const char *string, size_t *olen)
Parse RFC3986 encoded 'string' back to unescaped version.
char * **oauth_sign_hmac_sha1** (const char *m, const char *k)
returns base64 encoded HMAC-SHA1 signature for given message and key.
char * **oauth_sign_hmac_sha1_raw** (const char *m, const size_t ml, const char *k, const size_t kl)
same as **oauth_sign_hmac_sha1** *but allows one to specify length of message and key (in case they contain null chars).*
char * **oauth_sign_plaintext** (const char *m, const char *k)
returns plaintext signature for the given key.
char * **oauth_sign_rsa_sha1** (const char *m, const char *k)
returns RSA-SHA1 signature for given data.
int **oauth_verify_rsa_sha1** (const char *m, const char *c, const char *s)
verify RSA-SHA1 signature.
char * **oauth_catenc** (int len,...)
url-escape strings and concatenate with '&' separator.
int **oauth_split_url_parameters** (const char *url, char ***argv)
splits the given url into a parameter array.
int **oauth_split_post_paramters** (const char *url, char ***argv, short qesc)
splits the given url into a parameter array.
char * **oauth_serialize_url** (int argc, int start, char **argv)
build a url query string from an array.
char * **oauth_serialize_url_sep** (int argc, int start, char **argv, char *sep, int mod)

encode query parameters from an array.
char * **oauth_serialize_url_parameters** (int argc, char **argv)
build a query parameter string from an array.
char * **oauth_gen_nonce** ()
generate a random string between 15 and 32 chars length and return a pointer to it.
int **oauth_cmpstringp** (const void *p1, const void *p2)
string compare function for oauth parameters.
int **oauth_param_exists** (char **argv, int argc, char *key)
search array for parameter key.
void **oauth_add_param_to_array** (int *argcp, char ***argvp, const char *addparam)
add query parameter to array
void **oauth_free_array** (int *argcp, char ***argvp)
free array args
int **oauth_time_independent_equals_n** (const char *a, const char *b, size_t len_a, size_t len_b)
compare two strings in constant-time (as to not let an attacker guess how many leading chars are correct: http://rdist.root.org/2010/01/07/timing-independent-array-comparison/)
int **oauth_time_indepenent_equals_n** (const char *a, const char *b, size_t len_a, size_t len_b) attribute_deprecated
int **oauth_time_independent_equals** (const char *a, const char *b)
compare two strings in constant-time.
int **oauth_time_indepenent_equals** (const char *a, const char *b) attribute_deprecated
char * **oauth_sign_url2** (const char *url, char **postargs, **OAuthMethod** method, const char *http_method, const char *c_key, const char *c_secret, const char *t_key, const char *t_secret)
calculate OAuth-signature for a given HTTP request URL, parameters and oauth-tokens.
char * **oauth_sign_url** (const char *url, char **postargs, **OAuthMethod** method, const char *c_key, const char *c_secret, const char *t_key, const char *t_secret) attribute_deprecated
void **oauth_sign_array2_process** (int *argcp, char ***argvp, char **postargs, **OAuthMethod** method, const char *http_method, const char *c_key, const char *c_secret, const char *t_key, const char *t_secret)
the back-end behind by /ref oauth_sign_array2.
char * **oauth_sign_array2** (int *argcp, char ***argvp, char **postargs, **OAuthMethod** method, const char *http_method, const char *c_key, const char *c_secret, const char *t_key, const char *t_secret)
same as /ref oauth_sign_url with the url already split into parameter array
char * **oauth_sign_array** (int *argcp, char ***argvp, char **postargs, **OAuthMethod** method, const char *c_key, const char *c_secret, const char *t_key, const char *t_secret) attribute_deprecated
char * **oauth_body_hash_file** (char *filename)
calculate body hash (sha1sum) of given file and return a oauth_body_hash=xxxx parameter to be added to the request.
char * **oauth_body_hash_data** (size_t length, const char *data)
calculate body hash (sha1sum) of given data and return a oauth_body_hash=xxxx parameter to be added to the request.

char * **oauth_body_hash_encode** (size_t len, unsigned char *digest)
base64 encode digest, free it and return a URL parameter with the oauth_body_hash.
char * **oauth_sign_xmpp** (const char *xml, **OAuthMethod** method, const char *c_secret, const char *t_secret)
xep-0235 - TODO
char * **oauth_http_get** (const char *u, const char *q) attribute_deprecated
do a HTTP GET request, wait for it to finish and return the content of the reply.
char * **oauth_http_get2** (const char *u, const char *q, const char *customheader) attribute_deprecated
do a HTTP GET request, wait for it to finish and return the content of the reply.
char * **oauth_http_post** (const char *u, const char *p) attribute_deprecated
do a HTTP POST request, wait for it to finish and return the content of the reply.
char * **oauth_http_post2** (const char *u, const char *p, const char *customheader) attribute_deprecated
do a HTTP POST request, wait for it to finish and return the content of the reply.
char * **oauth_post_file** (const char *u, const char *fn, const size_t len, const char *customheader) attribute_deprecated
http post raw data from file.
char * **oauth_post_data** (const char *u, const char *data, size_t len, const char *customheader) attribute_deprecated
http post raw data the returned string needs to be freed by the caller (requires libcurl)
char * **oauth_post_data_with_callback** (const char *u, const char *data, size_t len, const char *customheader, void(*callback)(void *, int, size_t, size_t), void *callback_data) attribute_deprecated
http post raw data, with callback.
char * **oauth_send_data** (const char *u, const char *data, size_t len, const char *customheader, const char *httpMethod) attribute_deprecated
http send raw data.
char * **oauth_send_data_with_callback** (const char *u, const char *data, size_t len, const char *customheader, void(*callback)(void *, int, size_t, size_t), void *callback_data, const char *httpMethod) attribute_deprecated
http post raw data, with callback.

Detailed Description

OAuth.net implementation in POSIX-C.

Author:

Robin Gareus robin@gareus.org

Definition in file **oauth.h**.

Enumeration Type Documentation

enum OAuthMethod

signature method to used for signing the request.

Enumerator:

OA_HMAC
use HMAC-SHA1 request signing method

OA_RSA
use RSA signature

OA_PLAINTEXT
use plain text signature (for testing only)

Definition at line 66 of file oauth.h.

Function Documentation

void oauth_add_param_to_array (int * argcp, char *** argvp, const char * addparam)

add query parameter to array **Parameters:**

argcp pointer to array length int
argvp pointer to array values
addparam parameter to add (eg. ’foo=bar’)

char* oauth_body_hash_data (size_t length, const char * data)

calculate body hash (sha1sum) of given data and return a oauth_body_hash=xxxx parameter to be added to the request. The returned string needs to be freed by the calling function. The returned string is not yet url-escaped and suitable to be passed as argument to **oauth_catenc**.

see http://oauth.googlecode.com/svn/spec/ext/body_hash/1.0/oauth-bodyhash.html

Parameters:

length length of the data parameter in bytes
data to calculate the hash for

Returns:

URL oauth_body_hash parameter string

Examples:

tests/oauthbodyhash.c.

char* oauth_body_hash_encode (size_t len, unsigned char * digest)

base64 encode digest, free it and return a URL parameter with the oauth_body_hash. The returned hash needs to be freed by the calling function. The returned string is not yet url-escaped and thus suitable to be passed to **oauth_catenc**.

Parameters:

len length of the digest to encode
digest hash value to encode

Returns:

URL oauth_body_hash parameter string

char* oauth_body_hash_file (char * filename)

calculate body hash (sha1sum) of given file and return a oauth_body_hash=xxxx parameter to be added to the request. The returned string needs to be freed by the calling function.

see http://oauth.googlecode.com/svn/spec/ext/body_hash/1.0/oauth-bodyhash.html

Parameters:

filename the filename to calculate the hash for

Returns:

URL oauth_body_hash parameter string

Examples:

tests/oauthbodyhash.c.

char* oauth_catenc (int len, ...)

url-escape strings and concatenate with '&' separator. The number of strings to be concatenated must be given as first argument. all arguments thereafter must be of type (char *)

Parameters:

len the number of arguments to follow this parameter

Returns:

pointer to memory holding the concatenated strings - needs to be xfree(d) by the caller. or NULL in case we ran out of memory.

int oauth_cmpstringp (const void * p1, const void * p2)

string compare function for oauth parameters. used with qsort. needed to normalize request parameters. see http://oauth.net/core/1.0/#anchor14

Examples:

tests/oauthexample.c, **tests/oauthtest.c**, and **tests/oauthtest2.c**.

int oauth_decode_base64 (unsigned char * dest, const char * src)

Decode the base64 encoded string 'src' into the memory pointed to by 'dest'. **Parameters:**

dest Pointer to memory for holding the decoded string. Must be large enough to receive the decoded string.

src A base64 encoded string.

Returns:

the length of the decoded string if decode succeeded otherwise 0.

char* oauth_encode_base64 (int size, const unsigned char * src)

Base64 encode and return size data in 'src'. The caller must free the returned string.

Parameters:

size The size of the data in src

src The data to be base64 encode

Returns:

encoded string otherwise NULL

void oauth_free_array (int * argcp, char * argvp)**

free array args **Parameters:**

argcp pointer to array length int

argvp pointer to array values to be xfree()d

Examples:

tests/oauthtest2.c.

char* oauth_gen_nonce ()

generate a random string between 15 and 32 chars length and return a pointer to it. The value needs to be freed by the caller

Returns:

zero terminated random string.

char* oauth_http_get (const char * u, const char * q)

do a HTTP GET request, wait for it to finish and return the content of the reply. (requires libcurl or a command-line HTTP client)

If compiled **without** libcurl this function calls a command-line executable defined in the environment variable OAUTH_HTTP_GET_CMD - it defaults to curl -sA 'liboauth-agent/0.1' '%u' **where %u is replaced with the URL and query parameters.**

bash & wget example: export OAUTH_HTTP_CMD='wget -q -U 'liboauth-agent/0.1' '%u' '

WARNING: this is a tentative function. it's convenient and handy for testing or developing OAuth code. But don't rely on this function to become a stable part of this API. It does not do much error checking or handling for one thing..

NOTE: *u* and *q* are just concatenated with a '?' in between, unless *q* is NULL. in which case only *u* will be used.

Parameters:

u base url to get

q query string to send along with the HTTP request or NULL.

Returns:

In case of an error NULL is returned; otherwise a pointer to the replied content from HTTP server. latter needs to be freed by caller.

Deprecated

use libcurl - http://curl.haxx.se/libcurl/c/

Examples:

tests/oauthexample.c, and **tests/oauthtest.c**.

char* oauth_http_get2 (const char * u, const char * q, const char * customheader)

do a HTTP GET request, wait for it to finish and return the content of the reply. (requires libcurl)

This is equivalent to /ref oauth_http_get but allows one to specifiy a custom HTTP header andhas no support for commandline-curl.

If liboauth is compiled **without** libcurl this function always returns NULL.

Parameters:

u base url to get
q query string to send along with the HTTP request or NULL.
customheader specify custom HTTP header (or NULL for none) Multiple header elements can be passed separating them with ’\r\n’

Returns:

In case of an error NULL is returned; otherwise a pointer to the replied content from HTTP server. latter needs to be freed by caller.

Deprecated

use libcurl - http://curl.haxx.se/libcurl/c/

Examples:

tests/oauthtest2.c.

char* oauth_http_post (const char * u, const char * p)

do a HTTP POST request, wait for it to finish and return the content of the reply. (requires libcurl or a command-line HTTP client)

If compiled **without** libcurl this function calls a command-line executable defined in the environment variable OAUTH_HTTP_CMD - it defaults to curl -sA ’liboauth-agent/0.1’ -d ’%p’ ’%u’ **where %p is replaced with the postargs and %u is replaced with the URL.**

bash & wget example: export OAUTH_HTTP_CMD=’wget -q -U ’liboauth-agent/0.1’ &ndash;post-data=’%p’ ’%u’ ’

NOTE: This function uses the curl's default HTTP-POST Content-Type: application/x-www-form-urlencoded which is the only option allowed by oauth core 1.0 spec. Experimental code can use the Environment variable to transmit custom HTTP headers or parameters.

WARNING: this is a tentative function. it's convenient and handy for testing or developing OAuth code. But don't rely on this function to become a stable part of this API. It does not do much error checking for one thing..

Parameters:

u url to query
p postargs to send along with the HTTP request.

Returns:

replied content from HTTP server. needs to be freed by caller.

Deprecated

use libcurl - http://curl.haxx.se/libcurl/c/

Examples:

tests/oauthexample.c, and **tests/oauthtest.c**.

char* oauth_http_post2 (const char * u, const char * p, const char * customheader)

do a HTTP POST request, wait for it to finish and return the content of the reply. (requires libcurl)

It's equivalent to /ref oauth_http_post, but offers the possibility to specify a custom HTTP header and has no support for commandline-curl.

If liboauth is compiled **without** libcurl this function always returns NULL.

Parameters:

u url to query
p postargs to send along with the HTTP request.
customheader specify custom HTTP header (or NULL for none) Multiple header elements can be passed separating them with '\r\n'

Returns:

replied content from HTTP server. needs to be freed by caller.

Deprecated

use libcurl - http://curl.haxx.se/libcurl/c/

int oauth_param_exists (char ** argv, int argc, char * key)

search array for parameter key. **Parameters:**

argv length of array to search

argc parameter array to search

key key of parameter to check.

Returns:

FALSE (0) if array does not contain a parameter with given key, TRUE (1) otherwise.

char* oauth_post_data (const char * u, const char * data, size_t len, const char * customheader)

http post raw data the returned string needs to be freed by the caller (requires libcurl) see dislaimer: /ref oauth_http_post

Parameters:

u url to retrieve

data data to post

len length of the data in bytes.

customheader specify custom HTTP header (or NULL for default) Multiple header elements can be passed separating them with '\r\n'

Returns:

returned HTTP reply or NULL on error

Deprecated

use libcurl - http://curl.haxx.se/libcurl/c/

Examples:

tests/oauthbodyhash.c.

char* oauth_post_data_with_callback (const char * u, const char * data, size_t len, const char * customheader, void(*)(void *, int, size_t, size_t) callback, void * callback_data)

http post raw data, with callback. the returned string needs to be freed by the caller (requires libcurl)

Invokes the callback - in no particular order - when HTTP-request status updates occur. The callback is called with: void * callback_data: supplied on function call. int type: 0=data received, 1=data sent. size_t size: amount of data received or amount of data sent so far size_t totalsize: original amount of data to send, or amount of data received

Parameters:

u url to retrieve

data data to post along
len length of the file in bytes. set to ’0’ for autodetection
customheader specify custom HTTP header (or NULL for default) Multiple header elements can be passed separating them with ’\r\n’
callback specify the callback function
callback_data specify data to pass to the callback function

Returns:
returned HTTP reply or NULL on error

Deprecated
use libcurl - http://curl.haxx.se/libcurl/c/

char* oauth_post_file (const char * u, const char * fn, const size_t len, const char * customheader)
http post raw data from file. the returned string needs to be freed by the caller (requires libcurl)

see dislaimer: /ref oauth_http_post

Parameters:
u url to retrieve
fn filename of the file to post along
len length of the file in bytes. set to ’0’ for autodetection
customheader specify custom HTTP header (or NULL for default). Multiple header elements can be passed separating them with ’\r\n’

Returns:
returned HTTP reply or NULL on error

Deprecated
use libcurl - http://curl.haxx.se/libcurl/c/

char* oauth_send_data (const char * u, const char * data, size_t len, const char * customheader, const char * httpMethod)
http send raw data. similar to /ref oauth_http_post but provides for specifying the HTTP request method.

the returned string needs to be freed by the caller (requires libcurl)

see dislaimer: /ref oauth_http_post

Parameters:

u url to retrieve
data data to post
len length of the data in bytes.
customheader specify custom HTTP header (or NULL for default) Multiple header elements can be passed separating them with '\r\n'
httpMethod specify http verb ('GET'/'POST'/'PUT'/'DELETE') to be used. if httpMethod is NULL, a POST is executed.

Returns:

returned HTTP reply or NULL on error

Deprecated

use libcurl - http://curl.haxx.se/libcurl/c/

char* oauth_send_data_with_callback (const char * u, const char * data, size_t len, const char * customheader, void(*)(void *, int, size_t, size_t) callback, void * callback_data, const char * httpMethod)

http post raw data, with callback. the returned string needs to be freed by the caller (requires libcurl)

Invokes the callback - in no particular order - when HTTP-request status updates occur. The callback is called with: void * callback_data: supplied on function call. int type: 0=data received, 1=data sent. size_t size: amount of data received or amount of data sent so far size_t totalsize: original amount of data to send, or amount of data received

Parameters:

u url to retrieve
data data to post along
len length of the file in bytes. set to '0' for autodetection
customheader specify custom HTTP header (or NULL for default) Multiple header elements can be passed separating them with '\r\n'
callback specify the callback function
callback_data specify data to pass to the callback function
httpMethod specify http verb ('GET'/'POST'/'PUT'/'DELETE') to be used.

Returns:

returned HTTP reply or NULL on error

Deprecated

use libcurl - http://curl.haxx.se/libcurl/c/

char* oauth_serialize_url (int argc, int start, char ** argv)

build a url query string from an array. **Parameters:**

argc the total number of elements in the array
start element in the array at which to start concatenating.
argv parameter-array to concatenate.

Returns:

url string needs to be freed by the caller.

char* oauth_serialize_url_parameters (int argc, char ** argv)

build a query parameter string from an array. This function is a shortcut for **oauth_serialize_url** (argc, 1, argv). It strips the leading host/path, which is usually the first element when using oauth_split_url_parameters on an URL.

Parameters:

argc the total number of elements in the array
argv parameter-array to concatenate.

Returns:

url string needs to be freed by the caller.

char* oauth_serialize_url_sep (int argc, int start, char ** argv, char * sep, int mod)

encode query parameters from an array. **Parameters:**

argc the total number of elements in the array
start element in the array at which to start concatenating.
argv parameter-array to concatenate.
sep separator for parameters (usually ’&’)
mod - bitwise modifiers: 1: skip all values that start with ’oauth_’ 2: skip all values that don’t start with ’oauth_’ 4: double quotation marks are added around values (use with sep ’, ’ for HTTP Authorization header).

Returns:

url string needs to be freed by the caller.

Examples:

tests/oauthtest2.c.

char* oauth_sign_array (int * argcp, char * argvp, char ** postargs, OAuthMethod method, const char * c_key, const char * c_secret, const char * t_key, const char * t_secret)**

Deprecated

Use **oauth_sign_array2()** instead.

char* oauth_sign_array2 (int * argcp, char * argvp, char ** postargs, OAuthMethod method, const char * http_method, const char * c_key, const char * c_secret, const char * t_key, const char * t_secret)**
same as /ref oauth_sign_url with the url already split into parameter array **Parameters:**

argcp pointer to array length int
argvp pointer to array values (argv[0]='http://example.org:80/' argv[1]='first=QueryParamater' .. the array is modified: fi. oauth_ parameters are added) These arrays can be generated with /ref oauth_split_url_parameters or /ref oauth_split_post_paramters.
postargs This parameter points to an area where the return value is stored. If 'postargs' is NULL, no value is stored.
method specify the signature method to use. It is of type **OAuthMethod** and most likely **OA_HMAC**.
http_method The HTTP request method to use (ie 'GET', 'PUT',..) If NULL is given as 'http_method' this defaults to 'GET' when 'postargs' is also NULL and when postargs is not NULL 'POST' is used.
c_key consumer key
c_secret consumer secret
t_key token key
t_secret token secret

Returns:

the signed url or NULL if an error occurred.

void oauth_sign_array2_process (int * argcp, char * argvp, char ** postargs, OAuthMethod method, const char * http_method, const char * c_key, const char * c_secret, const char * t_key, const char * t_secret)**
the back-end behind by /ref oauth_sign_array2. however it does not serialize the signed URL again. The user needs to call /ref oauth_serialize_url (oA) and /ref oauth_free_array to do so.

This allows one to split parts of the URL to be used for OAuth HTTP Authorization header: see http://oauth.net/core/1.0a/#consumer_req_param the oauthtest2 example code does so.

Parameters:

argcp pointer to array length int
argvp pointer to array values (argv[0]='http://example.org:80/' argv[1]='first=QueryParamater' .. the array is modified: fi. oauth_ parameters are added) These arrays can be generated with /ref oauth_split_url_parameters or /ref oauth_split_post_paramters.
postargs This parameter points to an area where the return value is stored. If 'postargs' is NULL, no value is stored.

method specify the signature method to use. It is of type **OAuthMethod** and most likely **OA_HMAC**.
http_method The HTTP request method to use (ie ’GET’, ’PUT’,..) If NULL is given as ’http_method’ this defaults to ’GET’ when ’postargs’ is also NULL and when postargs is not NULL ’POST’ is used.
c_key consumer key
c_secret consumer secret
t_key token key
t_secret token secret

Returns:
void

Examples:
tests/oauthtest2.c.

char* oauth_sign_hmac_sha1 (const char * m, const char * k)
returns base64 encoded HMAC-SHA1 signature for given message and key. both data and key need to be urlencoded.

the returned string needs to be freed by the caller

Parameters:
m message to be signed
k key used for signing

Returns:
signature string.

Examples:
tests/selftest_wiki.c.

char* oauth_sign_hmac_sha1_raw (const char * m, const size_t ml, const char * k, const size_t kl)
same as **oauth_sign_hmac_sha1** but allows one to specify length of message and key (in case they contain null chars). **Parameters:**
m message to be signed
ml length of message
k key used for signing
kl length of key

Returns:

signature string.

char* oauth_sign_plaintext (const char * m, const char * k)

returns plaintext signature for the given key. the returned string needs to be freed by the caller

Parameters:

m message to be signed
k key used for signing

Returns:

signature string

char* oauth_sign_rsa_sha1 (const char * m, const char * k)

returns RSA-SHA1 signature for given data. the returned signature needs to be freed by the caller.

Parameters:

m message to be signed
k private-key PKCS and Base64-encoded

Returns:

base64 encoded signature string.

Examples:

tests/selftest_wiki.c.

char* oauth_sign_url (const char * url, char ** postargs, OAuthMethod method, const char * c_key, const char * c_secret, const char * t_key, const char * t_secret)

Deprecated

Use **oauth_sign_url2()** instead.

char* oauth_sign_url2 (const char * url, char ** postargs, OAuthMethod method, const char * http_method, const char * c_key, const char * c_secret, const char * t_key, const char * t_secret)

calculate OAuth-signature for a given HTTP request URL, parameters and oauth-tokens. if ’postargs’ is NULL a ’GET’ request is signed and the signed URL is returned. Else this fn will modify ’postargs’ to point to memory that contains the signed POST-variables and returns the base URL.

both, the return value and (if given) ’postargs’ need to be freed by the caller.

Parameters:

url The request URL to be signed. append all GET or POST query-parameters separated by either '?' or '&' to this parameter.
postargs This parameter points to an area where the return value is stored. If 'postargs' is NULL, no value is stored.
method specify the signature method to use. It is of type **OAuthMethod** and most likely **OA_HMAC**.
http_method The HTTP request method to use (ie 'GET', 'PUT',..) If NULL is given as 'http_method' this defaults to 'GET' when 'postargs' is also NULL and when postargs is not NULL 'POST' is used.
c_key consumer key
c_secret consumer secret
t_key token key
t_secret token secret

Returns:
the signed url or NULL if an error occurred.

Examples:
tests/oauthbodyhash.c, **tests/oauthexample.c**, and **tests/oauthtest.c**.

char* oauth_sign_xmpp (const char * xml, OAuthMethod method, const char * c_secret, const char * t_secret)
xep-0235 - TODO

int oauth_split_post_paramters (const char * url, char * argv, short qesc)**
splits the given url into a parameter array. (see **oauth_serialize_url** and **oauth_serialize_url_parameters** for the reverse)

Parameters:
url the url or query-string to parse.
argv pointer to a (char *) array where the results are stored. The array is re-allocated to match the number of parameters and each parameter-string is allocated with strdup. - The memory needs to be freed by the caller.
qesc use query parameter escape (vs post-param-escape) - if set to 1 all '+' are treated as spaces ' '

Returns:
number of parameter(s) in array.

int oauth_split_url_parameters (const char * url, char * argv)**
splits the given url into a parameter array. (see **oauth_serialize_url** and **oauth_serialize_url_parameters**

for the reverse) (see **oauth_split_post_paramters** for a more generic version)

Parameters:

url the url or query-string to parse; may be NULL
argv pointer to a (char *) array where the results are stored. The array is re-allocated to match the number of parameters and each parameter-string is allocated with strdup. - The memory needs to be freed by the caller.

Returns:

number of parameter(s) in array.

Examples:

tests/oauthexample.c, **tests/oauthtest.c**, and **tests/oauthtest2.c**.

int oauth_time_independent_equals (const char * a, const char * b)

compare two strings in constant-time. wrapper to **oauth_time_independent_equals_n** which calls strlen() for each argument.

Parameters:

a string to compare
b string to compare

returns 0 (false) if strings are not equal, and 1 (true) if strings are equal.

int oauth_time_independent_equals_n (const char * a, const char * b, size_t len_a, size_t len_b)

compare two strings in constant-time (as to not let an attacker guess how many leading chars are correct: http://rdist.root.org/2010/01/07/timing-independent-array-comparison/) **Parameters:**

a string to compare
b string to compare
len_a length of string a
len_b length of string b

returns 0 (false) if strings are not equal, and 1 (true) if strings are equal.

int oauth_time_indepenent_equals (const char * a, const char * b)

Deprecated

Use **oauth_time_independent_equals()** instead.

int oauth_time_indepenent_equals_n (const char * a, const char * b, size_t len_a, size_t len_b)

Deprecated

Use **oauth_time_independent_equals_n()** instead.

char* oauth_url_escape (const char * string)

Escape ’string’ according to RFC3986 and http://oauth.net/core/1.0/#encoding_parameters. **Parameters:**

string The data to be encoded

Returns:

encoded string otherwise NULL The caller must free the returned string.

char* oauth_url_unescape (const char * string, size_t * olen)

Parse RFC3986 encoded ’string’ back to unescaped version. **Parameters:**

string The data to be unescaped
olen unless NULL the length of the returned string is stored there.

Returns:

decoded string or NULL The caller must free the returned string.

int oauth_verify_rsa_sha1 (const char * m, const char * c, const char * s)

verify RSA-SHA1 signature. returns the output of EVP_VerifyFinal() for a given message, cert/pubkey and signature.

Parameters:

m message to be verified
c public-key or x509 certificate
s base64 encoded signature

Returns:

1 for a correct signature, 0 for failure and -1 if some other error occurred

Examples:

tests/selftest_wiki.c.

Author

Generated automatically by Doxygen for OAuth library functions from the source code.