Sagui library

v2.5.2

Generated by Doxygen 1.8.8

Tue Dec 10 2019 03:01:44

Contents

1 Main Page 1

2 Module Index 1

2.1 Modules . . . 1

3 Data Structure Index 1

3.1 Data Structures . . . 1

4 File Index 2

4.1 File List . . . 2

5 Module Documentation 3

5.1 API reference . . . 3

5.1.1 Detailed Description . . . 3

5.2 Utilities . . . 4

5.2.1 Detailed Description . . . 4

5.2.2 Typedef Documentation . . . 4

5.2.3 Function Documentation . . . 6

5.3 String . . . 11

5.3.1 Detailed Description . . . 11

5.3.2 Function Documentation . . . 11

5.4 String map . . . 14

5.4.1 Detailed Description . . . 14

5.4.2 Typedef Documentation . . . 14

5.4.3 Function Documentation . . . 15

5.5 HTTP server . . . 20

5.5.1 Detailed Description . . . 22

5.5.2 Macro Definition Documentation . . . 22

5.5.3 Typedef Documentation . . . 25

5.5.4 Function Documentation . . . 26

5.6 Path routing . . . 53

5.6.1 Detailed Description . . . 54

5.6.2 Typedef Documentation . . . 54

5.6.3 Function Documentation . . . 55

6 Data Structure Documentation 66

6.1 sg_entrypoint Struct Reference . . . 66

6.1.1 Detailed Description . . . 66

6.2 sg_entrypoints Struct Reference . . . 66

6.2.1 Detailed Description . . . 66

6.3 sg_httpauth Struct Reference . . . 66
6.3.1 Detailed Description . . . 66
6.4 sg_httpreq Struct Reference . . . 66
6.4.1 Detailed Description . . . 67
6.5 sg_httpres Struct Reference . . . 67
6.5.1 Detailed Description . . . 67
6.6 sg_httpsrv Struct Reference . . . 67
6.6.1 Detailed Description . . . 67
6.7 sg_httpupld Struct Reference . . . 67
6.7.1 Detailed Description . . . 68
6.8 sg_route Struct Reference . . . 68
6.8.1 Detailed Description . . . 68
6.9 sg_router Struct Reference . . . 68
6.9.1 Detailed Description . . . 68
6.10 sg_str Struct Reference . . . 68
6.10.1 Detailed Description . . . 69
6.11 sg_strmap Struct Reference . . . 69
6.11.1 Detailed Description . . . 69

7 File Documentation **69**

7.1 example_entrypoint.h File Reference . . . 69
7.2 example_httpauth.h File Reference . . . 69
7.3 example_httpcomp.h File Reference . . . 69
7.4 example_httpcookie.h File Reference . . . 69
7.5 example_httpreq_payload.h File Reference . . . 69
7.6 example_httpsrv.h File Reference . . . 69
7.7 example_httpsrv_benchmark.h File Reference . . . 69
7.8 example_httpsrv_tls.h File Reference . . . 69
7.9 example_httpsrv_tls_cert_auth.h File Reference . . . 69
7.10 example_httpupldsh File Reference . . . 69
7.11 example_router_segments.h File Reference . . . 69
7.12 example_router_simple.h File Reference . . . 70
7.13 example_router_srv.h File Reference . . . 70
7.14 example_router_vars.h File Reference . . . 70
7.15 example_str.h File Reference . . . 70
7.16 example_strmap.h File Reference . . . 70
7.17 sagui.h File Reference . . . 70
7.17.1 Macro Definition Documentation . . . 74

8 Example Documentation **74**

8.1 example_entrypoint.c . . . 74

8.2 example_httpauth.c . . . 75
8.3 example_httpcomp.c . . . 76
8.4 example_httpcookie.c . . . 77
8.5 example_httpreq_payload.c . . . 78
8.6 example_httpsrv.c . . . 79
8.7 example_httpsrv_benchmark.c . . . 80
8.8 example_httpsrv_tls.c . . . 81
8.9 example_httpsrv_tls_cert_auth.c . . . 82
8.10 example_httpuplds.c . . . 85
8.11 example_router_segments.c . . . 87
8.12 example_router_simple.c . . . 88
8.13 example_router_srv.c . . . 88
8.14 example_router_vars.c . . . 90
8.15 example_str.c . . . 91
8.16 example_strmap.c . . . 91

Index **93**

1 Main Page

- API reference

2 Module Index

2.1 Modules

Here is a list of all modules:

API reference **3**

Utilities **4**

String **11**

String map **14**

HTTP server **20**

Path routing **53**

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

sg_entrypoint **66**

sg_entrypoints 66

sg_httpauth 66

sg_httpreq 66

sg_httpres 67

sg_httpsrv 67

sg_httpupld 67

sg_route 68

sg_router 68

sg_str 68

sg_strmap 69

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

example_entrypoint.h 69

example_httpauth.h 69

example_httpcomp.h 69

example_httpcookie.h 69

example_httpreq_payload.h 69

example_httpsrv.h 69

example_httpsrv_benchmark.h 69

example_httpsrv_tls.h 69

example_httpsrv_tls_cert_auth.h 69

example_httpupld.h 69

example_router_segments.h 69

example_router_simple.h 70

example_router_srv.h 70

example_router_vars.h 70

example_str.h 70

example_strmap.h 70

sagui.h 70

5 Module Documentation

5.1 API reference

Modules

- Utilities
- String
- String map
- HTTP server
- Path routing

5.1.1 Detailed Description

The API reference grouped by feature.

5.2 Utilities

Typedefs

- typedef void *(* sg_malloc_func)(size_t size)
- typedef void *(* sg_realloc_func)(void *ptr, size_t size)
- typedef void(* sg_free_func)(void *ptr)
- typedef void(* sg_err_cb)(void *cls, const char *err)
- typedef ssize_t(* sg_write_cb)(void *handle, uint64_t offset, const char *buf, size_t size)
- typedef ssize_t(* sg_read_cb)(void *handle, uint64_t offset, char *buf, size_t size)
- typedef void(* sg_free_cb)(void *handle)
- typedef int(* sg_save_cb)(void *handle, bool overwritten)
- typedef int(* sg_save_as_cb)(void *handle, const char *path, bool overwritten)

Functions

- unsigned int sg_version (void)
- const char * sg_version_str (void)
- int sg_mm_set (sg_malloc_func malloc_func, sg_realloc_func realloc_func, sg_free_func free_func)
- void * sg_malloc (size_t size) __attribute__((malloc))
- void * sg_alloc (size_t size) __attribute__((malloc))
- void * sg_realloc (void *ptr, size_t size) __attribute__((malloc))
- void sg_free (void *ptr)
- char * sg_strerror (int errnum, char *errmsg, size_t errlen)
- bool sg_is_post (const char *method)
- char * sg_extract_entrypoint (const char *path)
- char * sg_tmpdir (void)
- ssize_t sg_eor (bool err)
- int sg_ip (const void *socket, char *buf, size_t size)

5.2.1 Detailed Description

All utility functions of the library.

5.2.2 Typedef Documentation

5.2.2.1 typedef void*(* sg_malloc_func)(size_t size)

Callback signature used to override the function which allocates a new memory space.

Parameters

in	*size*	Memory size to be allocated.

Returns

Pointer of the allocated memory.

Return values

NULL	If size is 0 or no memory space.

5.2.2.2 typedef void*(* sg_realloc_func)(void *ptr, size_t size)

Callback signature used to override the function which reallocates an existing memory block.

Parameters

in	*ptr*	Pointer of the memory to be reallocated.
in	*size*	Memory size to be reallocated.

Returns

Pointer of the reallocated memory.

5.2.2.3 typedef void(∗ sg_free_func)(void ∗ptr)

Callback signature used to override the function which frees a memory space previously allocated by sg_malloc(), sg_alloc() or sg_realloc().

Parameters

in	*ptr*	Pointer of the memory to be freed.

5.2.2.4 typedef void(∗ sg_err_cb)(void ∗cls, const char ∗err)

Callback signature used by functions that handle errors.

Parameters

out	*cls*	User-defined closure.
out	*err*	Error message.

5.2.2.5 typedef ssize_t(∗ sg_write_cb)(void ∗handle, uint64_t offset, const char ∗buf, size_t size)

Callback signature used by functions that write streams.

Parameters

out	*handle*	Stream handle.
out	*offset*	Current stream offset.
out	*buf*	Current buffer to be written.
out	*size*	Size of the current buffer to be written.

Returns

Total written buffer.

5.2.2.6 typedef ssize_t(∗ sg_read_cb)(void ∗handle, uint64_t offset, char ∗buf, size_t size)

Callback signature used by functions that read streams.

Parameters

out	*handle*	Stream handle.
out	*offset*	Current stream offset.
out	*buf*	Current read buffer.
out	*size*	Size of the current read buffer.

Returns

Total read buffer.

5.2.2.7 typedef void(∗ sg_free_cb)(void ∗handle)

Callback signature used by functions that free streams.

Parameters

out	*handle*	Stream handle.

5.2.2.8 typedef int(∗ sg_save_cb)(void ∗handle, bool overwritten)

Callback signature used by functions that save streams.

Parameters

out	*handle*	Stream handle.
out	*overwritten*	Overwrite an already existed stream.

Return values

0	Success.
E<ERROR>	User-defined error to abort the saving.

5.2.2.9 typedef int(∗ sg_save_as_cb)(void ∗handle, const char ∗path, bool overwritten)

Callback signature used by functions that save streams. It allows to specify the destination file path.

Parameters

out	*handle*	Stream handle.
out	*path*	Absolute path to store the stream.
out	*overwritten*	Overwrite an already existed stream.

Return values

0	Success.
E<ERROR>	User-defined error to abort the saving.

5.2.3 Function Documentation

5.2.3.1 unsigned int sg_version (void)

Returns the library version number.

Returns

Library version packed into a single integer.

5.2.3.2 const char∗ sg_version_str (void)

Returns the library version number as string in the format `<MAJOR>.<MINOR>.<PATCH>`.

Returns

Library version packed into a null-terminated string.

5.2.3.3 int sg_mm_set (sg_malloc_func *malloc_func,* sg_realloc_func *realloc_func,* sg_free_func *free_func*)

Overrides the standard functions `malloc(3)`, `realloc(3)` and `free(3)` set by default in the memory manager.

Parameters

in	*malloc_func*	Reference to override the function `malloc()`.
in	*realloc_func*	Reference to override the function `realloc()`.
in	*free_func*	Reference to override the function `free()`.

Return values

0	Success.
EINVAL	Invalid argument.

Note

It must be called before any other Sagui function or after all resources have been freed.

5.2.3.4 void* sg_malloc (size_t *size*)

Allocates a new memory space.

Parameters

in	*size*	Memory size to be allocated.

Returns

Pointer of the allocated memory.

Return values

NULL	If size is `0` or no memory space.

Note

Equivalent to `malloc(3)`.

5.2.3.5 void* sg_alloc (size_t *size*)

Allocates a new zero-initialized memory space.

Parameters

in	*size*	Memory size to be allocated.

Returns

Pointer of the zero-initialized allocated memory.

Return values

NULL	If size is `0` or no memory space.

Examples:

example_httpsrv_tls_cert_auth.c.

5.2.3.6 void* sg_realloc (void * *ptr*, size_t *size*)

Reallocates an existing memory block.

Parameters

in	*ptr*	Pointer of the memory to be reallocated.
in	*size*	Memory size to be reallocated.

Returns

Pointer of the reallocated memory.

Note

Equivalent to `realloc(3)`.

5.2.3.7 void sg_free (void * *ptr*)

Frees a memory space previously allocated by sg_malloc(), sg_alloc() or sg_realloc().

Parameters

in	*ptr*	Pointer of the memory to be freed.

Note

Equivalent to `free(3)`.

Examples:

example_httpsrv_tls_cert_auth.c.

5.2.3.8 char* sg_strerror (int *errnum*, char * *errmsg*, size_t *errlen*)

Returns string describing an error number.

Parameters

in	*errnum*	Error number.
in,out	*errmsg*	Pointer of a string to store the error message.
in	*errlen*	Length of the error message.

Returns

Pointer to **`str`**.

Examples:

example_httpsrv_tls_cert_auth.c, and example_httpupldsc.

5.2.3.9 bool sg_is_post (const char * *method*)

Checks if a string is a HTTP post method.

Parameters

in	*method*	Null-terminated string.

Return values

true	If **method** is POST, PUT, DELETE or OPTIONS.

5.2.3.10 char* sg_extract_entrypoint (const char * *path*)

Extracts the entry-point of a path or resource. For example, given a path `/api1/customer`, the part considered as entry-point is `/api1`.

Parameters

path	Path as null-terminated string.

Returns

Entry-point as null-terminated string.

Return values

NULL	If no memory space is available.

Warning

The caller must free the returned value.

5.2.3.11 char* sg_tmpdir (void)

Returns the system temporary directory.

Returns

Temporary directory as null-terminated string.

Return values

NULL	If no memory space is available.

Warning

The caller must free the returned value.

Examples:

example_httpuplds.c.

5.2.3.12 ssize_t sg_eor (bool *err*)

Indicates the end-of-read processed in sg_httpres_sendstream().

Parameters

in	*err*	`true` to return a value indicating a stream reading error.

Returns

Value to end a stream reading.

5.2.3.13 int sg_ip (const void * *socket*, char * *buf*, size_t *size*)

Obtains the IP of a socket handle (e.g. the one returned by sg_httpreq_client()) into a null-terminated string.

Parameters

in	*socket*	Socket handle.
out	*buf*	Pointer of the string to store the IP.
in	*size*	Size of the string to store the IP.

Return values

0	Success.
EINVAL	Invalid argument.
EAFNOSUPPORT	Address family not supported by protocol.
ENOSPC	No space left on device.

5.3 String

Data Structures

- struct sg_str

Functions

- struct sg_str * sg_str_new (void) __attribute__((malloc))
- void sg_str_free (struct sg_str *str)
- int sg_str_write (struct sg_str *str, const char *val, size_t len)
- int sg_str_printf_va (struct sg_str *str, const char *fmt, va_list ap)
- int sg_str_printf (struct sg_str *str, const char *fmt,...) __attribute__((format(printf
- int const char * sg_str_content (struct sg_str *str)
- size_t sg_str_length (struct sg_str *str)
- int sg_str_clear (struct sg_str *str)

5.3.1 Detailed Description

String handle and its related functions.

5.3.2 Function Documentation

5.3.2.1 struct sg_str* sg_str_new (void)

Creates a new zero-initialized string handle.

Returns

String handle.

Return values

NULL	If no memory space is available.

Examples:

example_httpuplds.c, example_router_srv.c, and example_str.c.

5.3.2.2 void sg_str_free (struct sg_str * *str*)

Frees the string handle previously allocated by sg_str_new().

Parameters

in	*str*	Pointer of the string handle to be freed.

Examples:

example_httpuplds.c, example_router_srv.c, and example_str.c.

5.3.2.3 int sg_str_write (struct sg_str * *str,* const char * *val,* size_t *len*)

Writes a null-terminated string to the string handle **`str`**. All strings previously written are kept.

Parameters

in	*str*	String handle.
in	*val*	String to be written.
in	*len*	Length of the string to be written.

Return values

0	Success.
EINVAL	Invalid argument.

5.3.2.4 int sg_str_printf_va (struct sg_str * *str*, const char * *fmt*, va_list *ap*)

Prints a null-terminated formatted string from the argument list to the string handle **str**.

Parameters

in	*str*	String handle.
in	*fmt*	Formatted string (following the same `printf()` format specification).
in	*ap*	Arguments list (handled by `va_start()`/ `va_end()`).

Return values

0	Success.
EINVAL	Invalid argument.

5.3.2.5 int sg_str_printf (struct sg_str * *str*, const char * *fmt*, ...)

Prints a null-terminated formatted string to the string handle **str**. All strings previously written are kept.

Parameters

in	*str*	String handle.
in	*fmt*	Formatted string (following the same `printf()` format specification).
in	...	Additional arguments (following the same `printf()` arguments specification).

Return values

0	Success.
EINVAL	Invalid argument.

Examples:

example_httpuplds.c, example_router_srv.c, and example_str.c.

5.3.2.6 int const char* sg_str_content (struct sg_str * *str*)

Returns the null-terminated string content from the string handle **str**.

Parameters

in	*str*	String handle.

Returns

Content as null-terminated string.

Return values

NULL	If the **str** is null and sets the `errno` to `EINVAL`.

Examples:

example_httpreq_payload.c, example_httpuplds.c, example_router_srv.c, and example_str.c.

5.3.2.7 size_t sg_str_length (struct sg_str * *str*)

Returns the total string length from the handle **str**.

Parameters

`in`	*str*	String handle.

Returns

Total string length.

Return values

EINVAL	Invalid argument.

5.3.2.8 int sg_str_clear (struct sg_str * *str*)

Clears all existing content in the string handle **str**.

Parameters

`in`	*str*	String handle.

Return values

0	Success.
EINVAL	Invalid argument.

Examples:

example_httpuplds.c.

5.4 String map

Data Structures

- struct sg_strmap

Typedefs

- typedef int(∗ sg_strmap_iter_cb)(void ∗cls, struct sg_strmap ∗pair)
- typedef int(∗ sg_strmap_sort_cb)(void ∗cls, struct sg_strmap ∗pair_a, struct sg_strmap ∗pair_b)

Functions

- const char ∗ sg_strmap_name (struct sg_strmap ∗pair)
- const char ∗ sg_strmap_val (struct sg_strmap ∗pair)
- int sg_strmap_add (struct sg_strmap ∗∗map, const char ∗name, const char ∗val)
- int sg_strmap_set (struct sg_strmap ∗∗map, const char ∗name, const char ∗val)
- int sg_strmap_find (struct sg_strmap ∗map, const char ∗name, struct sg_strmap ∗∗pair)
- const char ∗ sg_strmap_get (struct sg_strmap ∗map, const char ∗name)
- int sg_strmap_rm (struct sg_strmap ∗∗map, const char ∗name)
- int sg_strmap_iter (struct sg_strmap ∗map, sg_strmap_iter_cb cb, void ∗cls)
- int sg_strmap_sort (struct sg_strmap ∗∗map, sg_strmap_sort_cb cb, void ∗cls)
- unsigned int sg_strmap_count (struct sg_strmap ∗map)
- int sg_strmap_next (struct sg_strmap ∗∗next)
- void sg_strmap_cleanup (struct sg_strmap ∗∗map)

5.4.1 Detailed Description

String map handle and its related functions.

5.4.2 Typedef Documentation

5.4.2.1 typedef int(∗ sg_strmap_iter_cb)(void ∗cls, struct sg_strmap ∗pair)

Callback signature used by sg_strmap_iter() to iterate pairs of strings.

Parameters

out	*cls*	User-defined closure.
out	*pair*	Current iterated pair.

Return values

0	Success.
E<ERROR>	User-defined error to stop pairs iteration.

5.4.2.2 typedef int(∗ sg_strmap_sort_cb)(void ∗cls, struct sg_strmap ∗pair_a, struct sg_strmap ∗pair_b)

Callback signature used by sg_strmap_sort() to sort pairs of strings.

Parameters

out	*cls*	User-defined closure.
out	*pair_a*	Current left pair (A).
out	*pair_b*	Current right pair (B).

Return values

-1	A < B.
0	A == B.
1	A > B.

5.4.3 Function Documentation

5.4.3.1 const char* sg_strmap_name (struct sg_strmap * *pair*)

Returns a name from the **pair**.

Parameters

in	*pair*	Pair of name-value.

Returns

Name as null-terminated string.

Return values

NULL	If the **pair** is null and sets the errno to EINVAL.

Examples:

example_strmap.c.

5.4.3.2 const char* sg_strmap_val (struct sg_strmap * *pair*)

Returns a value from the **pair**.

Parameters

in	*pair*	Pair of name-value.

Returns

Value as null-terminated string.

Return values

NULL	If the **pair** is null and sets the errno to EINVAL.

Examples:

example_strmap.c.

5.4.3.3 int sg_strmap_add (struct sg_strmap ** *map*, const char * *name*, const char * *val*)

Adds a pair of name-value to the string **map**.

Parameters

in,out	*map*	Pairs map pointer to add a new pair.
in	*name*	Pair name.
in	*val*	Pair value.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.

Note

It cannot check if a name already exists in a pair added to the **map**, then the uniqueness must be managed by the application.

5.4.3.4 int sg_strmap_set (struct sg_strmap ** *map*, const char * *name*, const char * *val*)

Sets a pair of name-value to the string **map**.

Parameters

in,out	*map*	Pairs map pointer to set a new pair.
in	*name*	Pair name.
in	*val*	Pair value.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.

Note

If a name already exists in a pair previously added into the **map**, then the function replaces its value, otherwise it is added as a new pair.

Examples:

example_strmap.c.

5.4.3.5 int sg_strmap_find (struct sg_strmap * *map*, const char * *name*, struct sg_strmap ** *pair*)

Finds a pair by name.

Parameters

in	*map*	Pairs map.
in	*name*	Name to find the pair.
in,out	*pair*	Pointer of the variable to store the found pair.

Return values

0	Success.
EINVAL	Invalid argument.

ENOENT	Pair not found.
ENOMEM	Out of memory.

Examples:

example_strmap.c.

5.4.3.6 const char* sg_strmap_get (struct sg_strmap * *map*, const char * *name*)

Gets a pair by name and returns the value.

Parameters

in	*map*	Pairs map.
in	*name*	Name to get the pair.

Returns

Pair value as null-terminated string.

Return values

NULL	If **map** or **name** is null or pair is not found.

Examples:

example_httpcomp.c, example_httpcookie.c, and example_httpuplds.c.

5.4.3.7 int sg_strmap_rm (struct sg_strmap ** *map*, const char * *name*)

Removes a pair by name.

Parameters

in	*map*	Pointer to the pairs map.
in	*name*	Name to find and then remove the pair.

Return values

0	Success.
EINVAL	Invalid argument.
ENOENT	Pair already removed.
ENOMEM	Out of memory.

5.4.3.8 int sg_strmap_iter (struct sg_strmap * *map*, sg_strmap_iter_cb *cb*, void * *cls*)

Iterates over pairs map.

Parameters

in	*map*	Pairs map.
in	*cb*	Callback to iterate the pairs.
in,out	*cls*	User-specified value.

Return values

0	Success.
EINVAL	Invalid argument.

Returns

Callback result when it is different from `0`.

Examples:

example_strmap.c.

5.4.3.9 int sg_strmap_sort (struct sg_strmap ** *map*, sg_strmap_sort_cb *cb*, void * *cls*)

Sorts the pairs map.

Parameters

`in,out`	*map*	Pointer to the pairs map.
`in`	*cb*	Callback to sort the pairs.
`in,out`	*cls*	User-specified value.

Return values

0	Success.
EINVAL	Invalid argument.

Examples:

example_strmap.c.

5.4.3.10 unsigned int sg_strmap_count (struct sg_strmap * *map*)

Counts the total pairs in the map.

Parameters

`in`	*map*	Pairs map.

Returns

Total of pairs.

Return values

0	If the list is empty or null.

5.4.3.11 int sg_strmap_next (struct sg_strmap ** *next*)

Returns the next pair in the map.

Parameters

`in,out`	*next*	Pointer to the next pair.

Return values

0	Success.
EINVAL	Invalid argument.

5.4.3.12 void sg_strmap_cleanup (struct sg_strmap ** *map*)

Cleans the entire map.

Parameters

in,out	*map*	Pointer to the pairs map.

Examples:

example_strmap.c.

5.5 HTTP server

Data Structures

- struct sg_httpauth
- struct sg_httpupld
- struct sg_httpreq
- struct sg_httpres
- struct sg_httpsrv

Macros

- #define sg_httpres_send(res, val, content_type, status)
- #define sg_httpres_download(res, filename) sg_httpres_sendfile2((res), 0, 0, 0, (filename), "attachment", 200)
- #define sg_httpres_render(res, filename) sg_httpres_sendfile2((res), 0, 0, 0, (filename), "inline", 200)
- #define sg_httpres_zsend(res, val, content_type, status)
- #define sg_httpres_zdownload(res, filename) sg_httpres_zsendfile2((res), 1, 0, 0, 0, (filename), "attachment", 200)
- #define sg_httpres_zrender(res, filename) sg_httpres_zsendfile2((res), 1, 0, 0, 0, (filename), "inline", 200)

Typedefs

- typedef void(∗ sg_httpsrv_cli_cb)(void ∗cls, const void ∗client, bool ∗closed)
- typedef bool(∗ sg_httpauth_cb)(void ∗cls, struct sg_httpauth ∗auth, struct sg_httpreq ∗req, struct sg_httpres ∗res)
- typedef int(∗ sg_httpupld_cb)(void ∗cls, void ∗∗handle, const char ∗dir, const char ∗field, const char ∗name, const char ∗mime, const char ∗encoding)
- typedef int(∗ sg_httpuplds_iter_cb)(void ∗cls, struct sg_httpupld ∗upld)
- typedef void(∗ sg_httpreq_cb)(void ∗cls, struct sg_httpreq ∗req, struct sg_httpres ∗res)

Functions

- int sg_httpauth_set_realm (struct sg_httpauth ∗auth, const char ∗realm)
- const char ∗ sg_httpauth_realm (struct sg_httpauth ∗auth)
- int sg_httpauth_deny2 (struct sg_httpauth ∗auth, const char ∗reason, const char ∗content_type, unsigned int status)
- int sg_httpauth_deny (struct sg_httpauth ∗auth, const char ∗reason, const char ∗content_type)
- int sg_httpauth_cancel (struct sg_httpauth ∗auth)
- const char ∗ sg_httpauth_usr (struct sg_httpauth ∗auth)
- const char ∗ sg_httpauth_pwd (struct sg_httpauth ∗auth)
- int sg_httpuplds_iter (struct sg_httpupld ∗uplds, sg_httpuplds_iter_cb cb, void ∗cls)
- int sg_httpuplds_next (struct sg_httpupld ∗∗upld)
- unsigned int sg_httpuplds_count (struct sg_httpupld ∗uplds)
- void ∗ sg_httpupld_handle (struct sg_httpupld ∗upld)
- const char ∗ sg_httpupld_dir (struct sg_httpupld ∗upld)
- const char ∗ sg_httpupld_field (struct sg_httpupld ∗upld)
- const char ∗ sg_httpupld_name (struct sg_httpupld ∗upld)
- const char ∗ sg_httpupld_mime (struct sg_httpupld ∗upld)
- const char ∗ sg_httpupld_encoding (struct sg_httpupld ∗upld)
- uint64_t sg_httpupld_size (struct sg_httpupld ∗upld)
- int sg_httpupld_save (struct sg_httpupld ∗upld, bool overwritten)
- int sg_httpupld_save_as (struct sg_httpupld ∗upld, const char ∗path, bool overwritten)
- struct sg_strmap ∗∗ sg_httpreq_headers (struct sg_httpreq ∗req)

- struct sg_strmap ** sg_httpreq_cookies (struct sg_httpreq *req)
- struct sg_strmap ** sg_httpreq_params (struct sg_httpreq *req)
- struct sg_strmap ** sg_httpreq_fields (struct sg_httpreq *req)
- const char * sg_httpreq_version (struct sg_httpreq *req)
- const char * sg_httpreq_method (struct sg_httpreq *req)
- const char * sg_httpreq_path (struct sg_httpreq *req)
- struct sg_str * sg_httpreq_payload (struct sg_httpreq *req)
- bool sg_httpreq_is_uploading (struct sg_httpreq *req)
- struct sg_httpupld * sg_httpreq_uploads (struct sg_httpreq *req)
- const void * sg_httpreq_client (struct sg_httpreq *req)
- void * sg_httpreq_tls_session (struct sg_httpreq *req)
- int sg_httpreq_set_user_data (struct sg_httpreq *req, void *data)
- void * sg_httpreq_user_data (struct sg_httpreq *req)
- struct sg_strmap ** sg_httpres_headers (struct sg_httpres *res)
- int sg_httpres_set_cookie (struct sg_httpres *res, const char *name, const char *val)
- int sg_httpres_sendbinary (struct sg_httpres *res, void *buf, size_t size, const char *content_type, unsigned int status)
- int sg_httpres_sendfile2 (struct sg_httpres *res, uint64_t size, uint64_t max_size, uint64_t offset, const char *filename, const char *disposition, unsigned int status)
- int sg_httpres_sendfile (struct sg_httpres *res, uint64_t size, uint64_t max_size, uint64_t offset, const char *filename, bool downloaded, unsigned int status)
- int sg_httpres_sendstream (struct sg_httpres *res, uint64_t size, sg_read_cb read_cb, void *handle, sg_free_cb free_cb, unsigned int status)
- int sg_httpres_zsendbinary2 (struct sg_httpres *res, int level, void *buf, size_t size, const char *content_type, unsigned int status)
- int sg_httpres_zsendbinary (struct sg_httpres *res, void *buf, size_t size, const char *content_type, unsigned int status)
- int sg_httpres_zsendstream2 (struct sg_httpres *res, int level, uint64_t size, sg_read_cb read_cb, void *handle, sg_free_cb free_cb, unsigned int status)
- int sg_httpres_zsendstream (struct sg_httpres *res, sg_read_cb read_cb, void *handle, sg_free_cb free_cb, unsigned int status)
- int sg_httpres_zsendfile2 (struct sg_httpres *res, int level, uint64_t size, uint64_t max_size, uint64_t offset, const char *filename, const char *disposition, unsigned int status)
- int sg_httpres_zsendfile (struct sg_httpres *res, uint64_t size, uint64_t max_size, uint64_t offset, const char *filename, bool downloaded, unsigned int status)
- int sg_httpres_clear (struct sg_httpres *res)
- struct sg_httpsrv * sg_httpsrv_new2 (sg_httpauth_cb auth_cb, sg_httpreq_cb req_cb, sg_err_cb err_cb, void *cls) __attribute__((malloc))
- struct sg_httpsrv * sg_httpsrv_new (sg_httpreq_cb cb, void *cls) __attribute__((malloc))
- void sg_httpsrv_free (struct sg_httpsrv *srv)
- bool sg_httpsrv_tls_listen2 (struct sg_httpsrv *srv, const char *key, const char *pwd, const char *cert, const char *trust, const char *dhparams, uint16_t port, bool threaded)
- bool sg_httpsrv_tls_listen (struct sg_httpsrv *srv, const char *key, const char *cert, uint16_t port, bool threaded)
- bool sg_httpsrv_listen (struct sg_httpsrv *srv, uint16_t port, bool threaded)
- int sg_httpsrv_shutdown (struct sg_httpsrv *srv)
- uint16_t sg_httpsrv_port (struct sg_httpsrv *srv)
- bool sg_httpsrv_is_threaded (struct sg_httpsrv *srv)
- int sg_httpsrv_set_cli_cb (struct sg_httpsrv *srv, sg_httpsrv_cli_cb cb, void *cls)
- int sg_httpsrv_set_upld_cbs (struct sg_httpsrv *srv, sg_httpupld_cb cb, void *cls, sg_write_cb write_cb, sg_free_cb free_cb, sg_save_cb save_cb, sg_save_as_cb save_as_cb)
- int sg_httpsrv_set_upld_dir (struct sg_httpsrv *srv, const char *dir)
- const char * sg_httpsrv_upld_dir (struct sg_httpsrv *srv)
- int sg_httpsrv_set_post_buf_size (struct sg_httpsrv *srv, size_t size)
- size_t sg_httpsrv_post_buf_size (struct sg_httpsrv *srv)

- int sg_httpsrv_set_payld_limit (struct sg_httpsrv *srv, size_t limit)
- size_t sg_httpsrv_payld_limit (struct sg_httpsrv *srv)
- int sg_httpsrv_set_uplds_limit (struct sg_httpsrv *srv, uint64_t limit)
- uint64_t sg_httpsrv_uplds_limit (struct sg_httpsrv *srv)
- int sg_httpsrv_set_thr_pool_size (struct sg_httpsrv *srv, unsigned int size)
- unsigned int sg_httpsrv_thr_pool_size (struct sg_httpsrv *srv)
- int sg_httpsrv_set_con_timeout (struct sg_httpsrv *srv, unsigned int timeout)
- unsigned int sg_httpsrv_con_timeout (struct sg_httpsrv *srv)
- int sg_httpsrv_set_con_limit (struct sg_httpsrv *srv, unsigned int limit)
- unsigned int sg_httpsrv_con_limit (struct sg_httpsrv *srv)

5.5.1 Detailed Description

Fast event-driven HTTP server.

5.5.2 Macro Definition Documentation

5.5.2.1 #define sg_httpres_send(*res, val, content_type, status*)

Value:

```
sg_httpres_sendbinary((res), (void *) (val),                                \
                      (((val) != NULL) ? strlen((val)) : 0), (content_type), \
                      (status))
```

Sends a null-terminated string content to the client.

Parameters

in	*res*	Response handle.
in	*val*	Null-terminated string.
in	*content_type*	Content type.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
ENOMEM	Out of memory.

Examples:

example_httpauth.c, example_httpcookie.c, example_httpreq_payload.c, example_httpsrv.c, example_↩ httpsrv_benchmark.c, example_httpsrv_tls.c, example_httpsrv_tls_cert_auth.c, example_httpuplds.c, and example_router_srv.c.

5.5.2.2 #define sg_httpres_download(*res, filename*) sg_httpres_sendfile2((res), 0, 0, 0, (filename), "attachment", 200)

Offer a file as download.

Parameters

in	*res*	Response handle.

in	*filename*	Path of the file to be sent.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
EISDIR	Is a directory.
EBADF	Bad file number.
ENOMEM	Out of memory.

Examples:

example_httpuplds.c.

5.5.2.3 #define sg_httpres_render(*res, filename*) sg_httpres_sendfile2((res), 0, 0, 0, (filename), "inline", 200)

Sends a file to be rendered.

Parameters

in	*res*	Response handle.
in	*filename*	Path of the file to be sent.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
EISDIR	Is a directory.
EBADF	Bad file number.
ENOMEM	Out of memory.

5.5.2.4 #define sg_httpres_zsend(*res, val, content_type, status*)

Value:

```
sg_httpres_zsendbinary((res), (void *) (val),                                  \
                       (((val) != NULL) ? strlen((val)) : 0),                    \
                       (content_type), (status))
```

Compresses a null-terminated string content and sends it to the client. The compression is done by zlib library using the DEFLATE compression algorithm.

Parameters

in	*res*	Response handle.
in	*val*	Null-terminated string.
in	*content_type*	Content type.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.

ENOBUFS	No buffer space available.
EALREADY	Operation already in progress.
Z_<ERROR>	zlib error as negative number.

Note

When compression succeeds, the header `Content-Encoding: deflate` is automatically added to the response.

5.5.2.5 #define sg_httpres_zdownload(*res, filename*) sg_httpres_zsendfile2((res), 1, 0, 0, 0, (filename), "attachment", 200)

Compresses a file in Gzip format and offer it as download. The compression is done by zlib library using the DEFLATE compression algorithm.

Parameters

in	*res*	Response handle.
in	*filename*	Path of the file to be compressed and sent.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
EISDIR	Is a directory.
EBADF	Bad file number.
ENOMEM	Out of memory.
Z_<ERROR>	zlib error as negative number.

Note

When compression succeeds, the header `Content-Encoding: gzip` is automatically added to the response.

5.5.2.6 #define sg_httpres_zrender(*res, filename*) sg_httpres_zsendfile2((res), 1, 0, 0, 0, (filename), "inline", 200)

Compresses a file in Gzip formant and sends it to be rendered. The compression is done by zlib library using the DEFLATE compression algorithm.

Parameters

in	*res*	Response handle.
in	*filename*	Path of the file to be sent.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
EISDIR	Is a directory.
EBADF	Bad file number.
ENOMEM	Out of memory.
Z_<ERROR>	zlib error as negative number.

Note

When compression succeeds, the header `Content-Encoding: gzip` is automatically added to the response.

5.5.3 Typedef Documentation

5.5.3.1 typedef void(∗ sg_httpsrv_cli_cb)(void ∗cls, const void ∗client, bool ∗closed)

Callback signature used to handle client events.

Parameters

out	*cls*	User-defined closure.
out	*client*	Socket handle of the client.
in,out	*closed*	Indicates if the client is connected allowing to close it.

5.5.3.2 typedef bool(∗ sg_httpauth_cb)(void ∗cls, struct sg_httpauth ∗auth, struct sg_httpreq ∗req, struct sg_httpres ∗res)

Callback signature used to grant or deny the user access to the server resources.

Parameters

out	*cls*	User-defined closure.
out	*auth*	Authentication handle.
out	*req*	Request handle.
out	*res*	Response handle.

Return values

true	Grants the user access.
false	Denies the user access.

5.5.3.3 typedef int(∗ sg_httpupld_cb)(void ∗cls, void ∗∗handle, const char ∗dir, const char ∗field, const char ∗name, const char ∗mime, const char ∗encoding)

Callback signature used to handle uploaded files and/or fields.

Parameters

out	*cls*	User-defined closure.
in,out	*handle*	Stream handle pointer.
out	*dir*	Directory to store the uploaded files.
out	*field*	Posted field.
out	*name*	Uploaded file name.
out	*mime*	Uploaded file content-type (e.g.: `text/plain`, `image/png`, `application/json` etc.).
out	*encoding*	Uploaded file transfer-encoding (e.g.: `chunked`, `deflate`, `gzip` etc.).

Return values

0	Success.
E<ERROR>	User-defined error to refuse the upload.

5.5.3.4 typedef int(∗ sg_httpupldsiter_cb)(void ∗cls, struct sg_httpupld ∗upld)

Callback signature used to iterate uploaded files.

Parameters

out	*cls*	User-defined closure.

out	*upld*	Current upload item.

Return values

0	Success.
E<ERROR>	User-defined error to stop list iteration.

5.5.3.5 typedef void(* sg_httpreq_cb)(void *cls, struct sg_httpreq *req, struct sg_httpres *res)

Callback signature used to handle requests and responses.

Parameters

out	*cls*	User-defined closure.
out	*req*	Request handle.
out	*res*	Response handle.

5.5.4 Function Documentation

5.5.4.1 int sg_httpauth_set_realm (struct sg_httpauth * *auth*, const char * *realm*)

Sets the authentication protection space (realm).

Parameters

in	*auth*	Authentication handle.
in	*realm*	Realm string.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Realm already set.
ENOMEM	Out of memory.

Examples:

example_httpauth.c.

5.5.4.2 const char* sg_httpauth_realm (struct sg_httpauth * *auth*)

Gets the authentication protection space (realm).

Parameters

in	*auth*	Authentication handle.

Returns

Realm as null-terminated string.

Return values

NULL	If **auth** is null and sets the errno to EINVAL.

5.5.4.3 int sg_httpauth_deny2 (struct sg_httpauth * *auth*, const char * *reason*, const char * *content_type*, unsigned int *status*)

Deny the authentication sending the reason to the user.

Parameters

in	*auth*	Authentication handle.
in	*reason*	Denial reason.
in	*content_type*	Content type.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Already denied.
ENOMEM	Out of memory.

5.5.4.4 int sg_httpauth_deny (struct sg_httpauth * *auth*, const char * *reason*, const char * *content_type*)

Deny the authentication sending the reason to the user.

Parameters

in	*auth*	Authentication handle.
in	*reason*	Denial reason.
in	*content_type*	Content type.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Already denied.
ENOMEM	Out of memory.

Examples:

example_httpauth.c.

5.5.4.5 int sg_httpauth_cancel (struct sg_httpauth * *auth*)

Cancels the authentication loop while the user is trying to acess the server.

Parameters

in	*auth*	Authentication handle.

Return values

0	Success.
EINVAL	Invalid argument.

5.5.4.6 const char* sg_httpauth_usr (struct sg_httpauth * *auth*)

Returns the authentication user.

Parameters

in	*auth*	Authentication handle.

Returns

User as null-terminated string.

Return values

NULL	If **auth** is null and sets the `errno` to `EINVAL`.

Examples:

example_httpauth.c.

5.5.4.7 const char* sg_httpauth_pwd (struct sg_httpauth * *auth*)

Returns the authentication password.

Parameters

`in`	*auth*	Authentication handle.

Returns

Password as null-terminated string.

Return values

NULL	If **auth** is null and sets the `errno` to `EINVAL`.

Examples:

example_httpauth.c.

5.5.4.8 int sg_httpuplds_iter (struct sg_httpupld * *uplds*, sg_httpuplds_iter_cb *cb*, void * *cls*)

Iterates over all the upload items in the **uplds** list.

Parameters

`in`	*uplds*	Uploads list handle.
`in`	*cb*	Callback to iterate over upload items.
`in`	*cls*	User-defined closure.

Return values

0	Success.
EINVAL	Invalid argument.
E<ERROR>	User-defined error to abort the list iteration.

5.5.4.9 int sg_httpuplds_next (struct sg_httpupld ** *upld*)

Gets the next upload item starting from the first item pointer **upld**.

Parameters

`in,out`	*upld*	Next upload item starting from the first item pointer.

Return values

0	Success.
EINVAL	Invalid argument.

Examples:

example_httpuplds.c.

5.5.4.10 unsigned int sg_httpuplds_count (struct sg_httpupld * *uplds*)

Counts the total upload items in the list **uplds**.

Parameters

in	*uplds*	Uploads list.

Returns

Total of items.

Return values

0	If the list is empty or null.

5.5.4.11 void* sg_httpupld_handle (struct sg_httpupld * *upld*)

Returns the stream handle of the upload handle **upld**.

Parameters

in	*upld*	Upload handle.

Returns

Stream handle.

Return values

NULL	If **upld** is null and sets the errno to EINVAL.

5.5.4.12 const char* sg_httpupld_dir (struct sg_httpupld * *upld*)

Returns the directory of the upload handle **upld**.

Parameters

in	*upld*	Upload handle.

Returns

Upload directory as null-terminated string.

Return values

NULL	If **upld** is null and sets the errno to EINVAL.

5.5.4.13 const char* sg_httpupld_field (struct sg_httpupld * *upld*)

Returns the field of the upload handle **upld**.

Parameters

in	*upld*	Upload handle.

Returns

Upload field as null-terminated string.

Return values

NULL	If **upld** is null and sets the `errno` to `EINVAL`.

5.5.4.14 const char* sg_httpupld_name (struct sg_httpupld * *upld*)

Returns the name of the upload handle **upld**.

Parameters

`in`	*upld*	Upload handle.

Returns

Upload name as null-terminated string.

Return values

NULL	If **upld** is null and sets the `errno` to `EINVAL`.

Examples:

example_httpuplds.c.

5.5.4.15 const char* sg_httpupld_mime (struct sg_httpupld * *upld*)

Returns the MIME (content-type) of the upload.

Parameters

`in`	*upld*	Upload handle.

Returns

Upload MIME as null-terminated string.

Return values

NULL	If **upld** is null and sets the `errno` to `EINVAL`.

5.5.4.16 const char* sg_httpupld_encoding (struct sg_httpupld * *upld*)

Returns the encoding (transfer-encoding) of the upload.

Parameters

`in`	*upld*	Upload handle.

Returns

Upload encoding as null-terminated string.

Return values

NULL	If **upld** is null and sets the `errno` to `EINVAL`.

5.5.4.17 uint64_t sg_httpupld_size (struct sg_httpupld * *upld*)

Returns the size of the upload.

Parameters

in	*upld*	Upload handle.

Returns

Upload size into `uint64`. If **`upld`** is null, sets the `errno` to `EINVAL`.

5.5.4.18 int sg_httpupld_save (struct sg_httpupld * *upld*, bool *overwritten*)

Saves the uploaded file defining the destination path by upload name and directory.

Parameters

in	*upld*	Upload handle.
in	*overwritten*	Overwrite upload file if it exists.

Return values

0	Success.
EINVAL	Invalid argument.
EEXIST	File already exists (if **`overwritten`** is `false`).
EISDIR	Destination file is a directory.

Examples:

example_httpuplds.c.

5.5.4.19 int sg_httpupld_save_as (struct sg_httpupld * *upld*, const char * *path*, bool *overwritten*)

Saves the uploaded file allowing to define the destination path.

Parameters

in	*upld*	Upload handle.
in	*path*	Absolute destination path.
in	*overwritten*	Overwrite upload file if it exists.

Return values

0	Success.
EINVAL	Invalid argument.
EEXIST	File already exists (if **`overwritten`** is `true`).
EISDIR	Destination file is a directory.

5.5.4.20 struct sg_strmap** sg_httpreq_headers (struct sg_httpreq * *req*)

Returns the client headers into sg_strmap map.

Parameters

in	*req*	Request handle.

Returns

Reference to the client headers map.

Return values

NULL	If **req** is null and sets the errno to EINVAL

Note

The headers map is automatically freed by the library.

Examples:

example_httpcomp.c.

5.5.4.21 struct sg_strmap** sg_httpreq_cookies (struct sg_httpreq * *req*)

Returns the client cookies into sg_strmap map.

Parameters

in	*req*	Request handle.

Returns

Reference to the client cookies map.

Return values

NULL	If **req** is null and sets the errno to EINVAL

Note

The cookies map is automatically freed by the library.

Examples:

example_httpcookie.c.

5.5.4.22 struct sg_strmap** sg_httpreq_params (struct sg_httpreq * *req*)

Returns the query-string into sg_strmap map.

Parameters

in	*req*	Request handle.

Returns

Reference to the query-string map.

Return values

NULL	If **req** is null and sets the errno to EINVAL

Note

The query-string map is automatically freed by the library.

Examples:

example_httpuplds.c.

5.5.4.23 struct sg_strmap** sg_httpreq_fields (struct sg_httpreq * *req*)

Returns the fields of a HTML form into sg_strmap map.

Parameters

in	*req*	Request handle.

Returns

Reference to the form fields map.

Return values

NULL	If **req** is null and sets the errno to EINVAL

Note

The form fields map is automatically freed by the library.

5.5.4.24 const char* sg_httpreq_version (struct sg_httpreq * *req*)

Returns the HTTP version.

Parameters

in	*req*	Request handle.

Returns

HTTP version as null-terminated string.

Return values

NULL	If **req** is null and sets the errno to EINVAL.

5.5.4.25 const char* sg_httpreq_method (struct sg_httpreq * *req*)

Returns the HTTP method.

Parameters

in	*req*	Request handle.

Returns

HTTP method as null-terminated string.

Return values

NULL	If **req** is null and sets the errno to EINVAL.

5.5.4.26 const char* sg_httpreq_path (struct sg_httpreq * *req*)

Returns the path component.

Parameters

in	*req*	Request handle.

Returns

Path component as null-terminated string.

Return values

NULL	If **req** is null and sets the errno to EINVAL.

Examples:

example_httpcookie.c, and example_router_srv.c.

5.5.4.27 struct sg_str* sg_httpreq_payload (struct sg_httpreq * *req*)

Returns the posting payload into a sg_str instance.

Parameters

in	*req*	Request handle.

Returns

Instance of the payload.

Return values

NULL	If **req** is null and sets the errno to EINVAL.

Note

The form payload instance is automatically freed by the library.

Examples:

example_httpreq_payload.c.

5.5.4.28 bool sg_httpreq_is_uploading (struct sg_httpreq * *req*)

Checks if the client is uploading data.

Parameters

in	*req*	Request handle.

Return values

true	If the client is uploading data, false otherwise. If **req** is null, sets the errno to EINVAL.

Examples:

example_httpuplds.c.

5.5.4.29 struct sg_httpupld* sg_httpreq_uploads (struct sg_httpreq * *req*)

Returns the list of the uploaded files.

Parameters

in	*req*	Request handle.

Returns

List of the uploaded files.

Return values

NULL	If **req** is null and sets the errno to EINVAL.

Note

The uploads list is automatically freed by the library.

Examples:

example_httpuplds.c.

5.5.4.30 const void* sg_httpreq_client (struct sg_httpreq * *req*)

Gets the socket handle of the client.

Parameters

in	*req*	Request handle.

Returns

Socket address of the client.

Return values

NULL	If **req** is null and sets the errno to EINVAL.

5.5.4.31 void* sg_httpreq_tls_session (struct sg_httpreq * *req*)

Returns the GnuTLS session handle.

Parameters

in	*req*	Request handle.

Return values

0	Success.
EINVAL	Invalid argument.

Examples:

example_httpsrv_tls_cert_auth.c.

5.5.4.32 int sg_httpreq_set_user_data (struct sg_httpreq * *req*, void * *data*)

Sets user data to the request handle.

Parameters

in	*req*	Request handle.
in	*data*	User data pointer.

Return values

0	Success.

EINVAL	Invalid argument.

5.5.4.33 void* sg_httpreq_user_data (struct sg_httpreq * *req*)

Gets user data from the request handle.

Parameters

in	*req*	Request handle.

Returns

User data pointer.

Return values

NULL	If **req** is null and sets the errno to EINVAL.

5.5.4.34 struct sg_strmap** sg_httpres_headers (struct sg_httpres * *res*)

Returns the server headers into sg_strmap map.

Parameters

in	*res*	Response handle.

Returns

Reference to the server headers map.

Return values

NULL	If **res** is null and sets the errno to EINVAL

Note

The headers map is automatically freed by the library.

5.5.4.35 int sg_httpres_set_cookie (struct sg_httpres * *res*, const char * *name*, const char * *val*)

Sets server cookie to the response handle.

Parameters

in	*res*	Response handle.
in	*name*	Cookie name.
in	*val*	Cookie value.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.

Examples:

example_httpcookie.c.

5.5.4.36 int sg_httpres_sendbinary (struct sg_httpres * *res*, void * *buf*, size_t *size*, const char * *content_type*, unsigned int *status*)

Sends a binary content to the client.

Parameters

in	*res*	Response handle.
in	*buf*	Binary content.
in	*size*	Content size.
in	*content_type*	Content type.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
ENOMEM	Out of memory.

Examples:

example_httpcomp.c.

5.5.4.37 int sg_httpres_sendfile2 (struct sg_httpres * *res*, uint64_t *size*, uint64_t *max_size*, uint64_t *offset*, const char * *filename*, const char * *disposition*, unsigned int *status*)

Sends a file to the client.

Parameters

in	*res*	Response handle.
in	*size*	Size of the file to be sent. Use zero to calculate automatically.
in	*max_size*	Maximum allowed file size. Use zero for no limit.
in	*offset*	Offset to start reading from in the file to be sent.
in	*filename*	Path of the file to be sent.
in	*disposition*	Content disposition as null-terminated string (attachment or inline).
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
EISDIR	Is a directory.
EBADF	Bad file number.
EFBIG	File too large.
ENOMEM	Out of memory.

Warning

The parameter `disposition` is not checked internally, thus any non-`NULL` value is passed directly to the header `Content-Disposition`.

5.5.4.38 int sg_httpres_sendfile (struct sg_httpres * *res*, uint64_t *size*, uint64_t *max_size*, uint64_t *offset*, const char * *filename*, bool *downloaded*, unsigned int *status*)

Sends a file to the client.

Parameters

in	*res*	Response handle.
in	*size*	Size of the file to be sent. Use zero to calculate automatically.
in	*max_size*	Maximum allowed file size. Use zero for no limit.
in	*offset*	Offset to start reading from in the file to be sent.
in	*filename*	Path of the file to be sent.
in	*downloaded*	If `true` it offer the file as download.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
EISDIR	Is a directory.
EBADF	Bad file number.
EFBIG	File too large.
ENOMEM	Out of memory.

5.5.4.39 int sg_httpres_sendstream (struct sg_httpres * *res*, uint64_t *size*, sg_read_cb *read_cb*, void * *handle*, sg_free_cb *free_cb*, unsigned int *status*)

Sends a stream to the client.

Parameters

in	*res*	Response handle.
in	*size*	Size of the stream.
in	*read_cb*	Callback to read data from stream handle.
in	*handle*	Stream handle.
in	*free_cb*	Callback to free the stream handle.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
ENOMEM	Out of memory.

Note

Use `size = 0` if the stream size is unknown.

5.5.4.40 int sg_httpres_zsendbinary2 (struct sg_httpres * *res*, int *level*, void * *buf*, size_t *size*, const char * *content_type*, unsigned int *status*)

Compresses a binary content and sends it to the client. The compression is done by zlib library using the DEFLATE compression algorithm.

Parameters

in	*res*	Response handle.
in	*level*	Compression level (1..9 or -1 for default).
in	*buf*	Binary content.
in	*size*	Content size.

in	*content_type*	Content type.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.
ENOBUFS	No buffer space available.
EALREADY	Operation already in progress.
Z_<ERROR>	zlib error as negative number.

Note

When compression succeeds, the header `Content-Encoding: deflate` is automatically added to the response.

5.5.4.41 int sg_httpres_zsendbinary (struct sg_httpres * *res*, void * *buf*, size_t *size*, const char * *content_type*, unsigned int *status*)

Compresses a binary content and sends it to the client. The compression is done by zlib library using the DEFLATE compression algorithm.

Parameters

in	*res*	Response handle.
in	*buf*	Binary content.
in	*size*	Content size.
in	*content_type*	Content type.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.
ENOBUFS	No buffer space available.
EALREADY	Operation already in progress.
Z_<ERROR>	zlib error as negative number.

Note

When compression succeeds, the header `Content-Encoding: deflate` is automatically added to the response.

Examples:

example_httpcomp.c.

5.5.4.42 int sg_httpres_zsendstream2 (struct sg_httpres * *res*, int *level*, uint64_t *size*, sg_read_cb *read_cb*, void * *handle*, sg_free_cb *free_cb*, unsigned int *status*)

Compresses a stream and sends it to the client. The compression is done by zlib library using the DEFLATE compression algorithm.

Parameters

in	*res*	Response handle.
in	*level*	Compression level (1..9 or -1 for default).
in	*size*	Size of the stream.
in	*read_cb*	Callback to read data from stream handle.
in	*handle*	Stream handle.
in	*free_cb*	Callback to free the stream handle.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
ENOMEM	Out of memory.
Z_<ERROR>	zlib error as negative number.

Note

When compression succeeds, the header `Content-Encoding: deflate` is automatically added to the response.

5.5.4.43 int sg_httpres_zsendstream (struct sg_httpres * *res*, sg_read_cb *read_cb*, void * *handle*, sg_free_cb *free_cb*, unsigned int *status*)

Compresses a stream and sends it to the client. The compression is done by zlib library using the DEFLATE compression algorithm.

Parameters

in	*res*	Response handle.
in	*read_cb*	Callback to read data from stream handle.
in	*handle*	Stream handle.
in	*free_cb*	Callback to free the stream handle.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
ENOMEM	Out of memory.
Z_<ERROR>	zlib error as negative number.

Note

When compression succeeds, the header `Content-Encoding: deflate` is automatically added to the response.

5.5.4.44 int sg_httpres_zsendfile2 (struct sg_httpres * *res*, int *level*, uint64_t *size*, uint64_t *max_size*, uint64_t *offset*, const char * *filename*, const char * *disposition*, unsigned int *status*)

Compresses a file in Gzip format and sends it to the client. The compression is done by zlib library using the DEFLATE compression algorithm.

Parameters

in	*res*	Response handle.
in	*level*	Compression level (1..9 or -1 for default).
in	*size*	Size of the file to be sent. Use zero to calculate automatically.
in	*max_size*	Maximum allowed file size. Use zero for no limit.
in	*offset*	Offset to start reading from in the file to be sent.
in	*filename*	Path of the file to be sent.
in	*disposition*	Content disposition as null-terminated string (attachment or inline).
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
EISDIR	Is a directory.
EBADF	Bad file number.
EFBIG	File too large.
ENOMEM	Out of memory.
Z_<ERROR>	zlib error as negative number.

Note

When compression succeeds, the header `Content-Encoding: gzip` is automatically added to the response.

Warning

The parameter `disposition` is not checked internally, thus any non-`NULL` value is passed directly to the header `Content-Disposition`.

5.5.4.45 int sg_httpres_zsendfile (struct sg_httpres * *res*, uint64_t *size*, uint64_t *max_size*, uint64_t *offset*, const char * *filename*, bool *downloaded*, unsigned int *status*)

Compresses a file in Gzip format and sends it to the client. The compression is done by zlib library using the DEFLATE compression algorithm.

Parameters

in	*res*	Response handle.
in	*size*	Size of the file to be sent. Use zero to calculate automatically.
in	*max_size*	Maximum allowed file size. Use zero for no limit.
in	*offset*	Offset to start reading from in the file to be sent.
in	*filename*	Path of the file to be sent.
in	*downloaded*	If `true` it offer the file as download.
in	*status*	HTTP status code.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Operation already in progress.
EISDIR	Is a directory.

EBADF	Bad file number.
EFBIG	File too large.
ENOMEM	Out of memory.
Z_<ERROR>	zlib error as negative number.

Note

When compression succeeds, the header `Content-Encoding: gzip` is automatically added to the response.

5.5.4.46 int sg_httpres_clear (struct sg_httpres * *res*)

Clears all headers, cookies, statuses and internal buffers of the response handle.

Parameters

`in`	*res*	Response handle.

Return values

0	Success.
EINVAL	Invalid argument.

5.5.4.47 struct sg_httpsrv* sg_httpsrv_new2 (sg_httpauth_cb *auth_cb*, sg_httpreq_cb *req_cb*, sg_err_cb *err_cb*, void * *cls*)

Creates a new HTTP server handle.

Parameters

`in`	*auth_cb*	Callback to grant/deny user access to the server resources.
`in`	*req_cb*	Callback to handle requests and responses.
`in`	*err_cb*	Callback to handle server errors.
`in`	*cls*	User-defined closure.

Returns

New HTTP server handle.

Return values

NULL	If no memory space is available.
NULL	If the **`req_cb`** or **`err_cb`** is null and sets the `errno` to `EINVAL`.

Examples:

example_httpauth.c.

5.5.4.48 struct sg_httpsrv* sg_httpsrv_new (sg_httpreq_cb *cb*, void * *cls*)

Creates a new HTTP server handle.

Parameters

`in`	*cb*	Callback to handle requests and responses.
`in`	*cls*	User-defined closure.

Returns

New HTTP server handle.

Return values

NULL	If the **cb** is null and sets the `errno` to `EINVAL`.

Examples:

example_httpcomp.c, example_httpcookie.c, example_httpreq_payload.c, example_httpsrv.c, example_↩ httpsrv_benchmark.c, example_httpsrv_tls.c, example_httpsrv_tls_cert_auth.c, example_httpuplds.c, and example_router_srv.c.

5.5.4.49 void sg_httpsrv_free (struct sg_httpsrv * *srv*)

Frees the server handle previously allocated by sg_httpsrv_new() or sg_httpsrv_new2().

Parameters

`in`	*srv*	Pointer of the server to be freed.

Note

If the server is running it stops before being freed.

Examples:

example_httpauth.c, example_httpcomp.c, example_httpcookie.c, example_httpreq_payload.c, example_↩ httpsrv.c, example_httpsrv_benchmark.c, example_httpsrv_tls.c, example_httpsrv_tls_cert_auth.c, example↩ _httpuplds.c, and example_router_srv.c.

5.5.4.50 bool sg_httpsrv_tls_listen2 (struct sg_httpsrv * *srv*, const char * *key*, const char * *pwd*, const char * *cert*, const char * *trust*, const char * *dhparams*, uint16_t *port*, bool *threaded*)

Starts the HTTPS server.

Parameters

`in`	*srv*	Server handle.
`in`	*key*	Memory pointer for the private key (key.pem) to be used by the HTTPS server.
`in`	*pwd*	Password for the private key.
`in`	*cert*	Memory pointer for the certificate (cert.pem) to be used by the HTTPS server.
`in`	*trust*	Memory pointer for the certificate (ca.pem) to be used by the HTTPS server for client authentication.
`in`	*dhparams*	Memory pointer for the Diffie Hellman parameters (dh.pem) to be used by the HTTPS server for key exchange.
`in`	*port*	Port for listening to connections.
`in`	*threaded*	Enables/disables the threaded model. If `true`, the server creates one thread per connection.

Return values

true	If the server is started, `false` otherwise. If **srv** is null, sets the `errno` to `EINVAL`.

Note

If port is `0`, the operating system will assign an unused port randomly.

Examples:

example_httpsrv_tls_cert_auth.c.

5.5.4.51 bool sg_httpsrv_tls_listen (struct sg_httpsrv * *srv*, const char * *key*, const char * *cert*, uint16_t *port*, bool *threaded*)

Starts the HTTPS server.

Parameters

in	*srv*	Server handle.
in	*key*	Memory pointer for the private key (key.pem) to be used by the HTTPS server.
in	*cert*	Memory pointer for the certificate (cert.pem) to be used by the HTTPS server.
in	*port*	Port for listening to connections.
in	*threaded*	Enables/disables the threaded model. If `true`, the server creates one thread per connection.

Return values

true	If the server is started, `false` otherwise. If **`srv`** is null, sets the `errno` to `EINVAL`.

Note

If port is `0`, the operating system will assign an unused port randomly.

Examples:

example_httpsrv_tls.c.

5.5.4.52 bool sg_httpsrv_listen (struct sg_httpsrv * *srv*, uint16_t *port*, bool *threaded*)

Starts the HTTP server.

Parameters

in	*srv*	Server handle.
in	*port*	Port for listening to connections.
in	*threaded*	Enables/disables the threaded model. If `true`, the server creates one thread per connection.

Return values

true	If the server is started, `false` otherwise. If **`srv`** is null, sets the `errno` to `EINVAL`.

Note

If port is `0`, the operating system will assign randomly an unused port.

Examples:

example_httpauth.c, example_httpcomp.c, example_httpcookie.c, example_httpreq_payload.c, example_↩ httpsrv.c, example_httpsrv_benchmark.c, example_httpuplds.c, and example_router_srv.c.

5.5.4.53 int sg_httpsrv_shutdown (struct sg_httpsrv * *srv*)

Stops the server not to accept new connections.

Parameters

in	*srv*	Server handle.

Return values

0	If the server is stopped. If **srv** is null, sets the errno to EINVAL.

Note

When sg_httpsrv_set_con_timeout() is set, the server waits for the clients to be closed before shutting down.

5.5.4.54 uint16_t sg_httpsrv_port (struct sg_httpsrv * *srv*)

Returns the server listening port.

Parameters

in	*srv*	Server handle.

Returns

Server listening port, 0 otherwise. If **srv** is null, sets the errno to EINVAL.

Examples:

example_httpauth.c, example_httpcomp.c, example_httpcookie.c, example_httpreq_payload.c, example_↩ httpsrv.c, example_httpsrv_benchmark.c, example_httpsrv_tls.c, example_httpsrv_tls_cert_auth.c, example↩ _httpuplds.c, and example_router_srv.c.

5.5.4.55 bool sg_httpsrv_is_threaded (struct sg_httpsrv * *srv*)

Checks if the server was started in threaded model.

Parameters

in	*srv*	Server handle.

Return values

true	If the server is in threaded model, false otherwise. If **srv** is null, sets the errno to EINVAL.

5.5.4.56 int sg_httpsrv_set_cli_cb (struct sg_httpsrv * *srv,* sg_httpsrv_cli_cb *cb,* void * *cls*)

Sets the server callback for client events.

Parameters

in	*srv*	Server handle.
in	*cb*	Callback to handle client events.
in	*cls*	User-defined closure.

Return values

0	Success.
EINVAL	Invalid argument.

5.5.4.57 int sg_httpsrv_set_upld_cbs (struct sg_httpsrv * *srv,* sg_httpupld_cb *cb,* void * *cls,* sg_write_cb *write_cb,* sg_free_cb *free_cb,* sg_save_cb *save_cb,* sg_save_as_cb *save_as_cb*)

Sets the server uploading callbacks.

Parameters

in	*srv*	Server handle.
in	*cb*	Callback to handle uploaded files and/or fields.
in	*cls*	User-defined closure.
in	*write_cb*	Callback to write the stream of the uploaded files.
in	*free_cb*	Callback to free stream of the uploaded files.
in	*save_cb*	Callback to save the uploaded files.
in	*save_as_cb*	Callback to save the uploaded files defining their path.

Return values

0	Success.
EINVAL	Invalid argument.

5.5.4.58 int sg_httpsrv_set_upld_dir (struct sg_httpsrv * *srv*, const char * *dir*)

Sets the directory to save the uploaded files.

Parameters

in	*srv*	Server handle.
in	*dir*	Directory as null-terminated string.

Return values

0	Success.
EINVAL	Invalid argument.

5.5.4.59 const char* sg_httpsrv_upld_dir (struct sg_httpsrv * *srv*)

Gets the directory of the uploaded files.

Parameters

in	*srv*	Server handle.

Returns

Directory as null-terminated string.

Return values

NULL	If the **srv** is null and sets the `errno` to `EINVAL`.

5.5.4.60 int sg_httpsrv_set_post_buf_size (struct sg_httpsrv * *srv*, size_t *size*)

Sets a size to the post buffering.

Parameters

in	*srv*	Server handle.
in	*size*	Post buffering size.

Return values

0	Success.

EINVAL	Invalid argument.

5.5.4.61 size_t sg_httpsrv_post_buf_size (struct sg_httpsrv * *srv*)

Gets the size of the post buffering.

Parameters

in	*srv*	Server handle.

Returns

Post buffering size.

Return values

0	If the **srv** is null and sets the errno to EINVAL.

5.5.4.62 int sg_httpsrv_set_payld_limit (struct sg_httpsrv * *srv*, size_t *limit*)

Sets a limit to the total payload.

Parameters

in	*srv*	Server handle.
in	*limit*	Payload total limit. Use zero for no limit.

Return values

0	Success.
EINVAL	Invalid argument.

5.5.4.63 size_t sg_httpsrv_payld_limit (struct sg_httpsrv * *srv*)

Gets the limit of the total payload.

Parameters

in	*srv*	Server handle.

Returns

Payload total limit.

Return values

0	If the **srv** is null and sets the errno to EINVAL.

5.5.4.64 int sg_httpsrv_set_uplds_limit (struct sg_httpsrv * *srv*, uint64_t *limit*)

Sets a limit to the total uploads.

Parameters

in	*srv*	Server handle.
in	*limit*	Uploads total limit. Use zero for no limit.

Return values

0	Success.
EINVAL	Invalid argument.

5.5.4.65 uint64_t sg_httpsrv_uplds_limit (struct sg_httpsrv * *srv*)

Gets the limit of the total uploads.

Parameters

in	*srv*	Server handle.

Returns

Uploads total limit.

Return values

0	If the **srv** is null and sets the errno to EINVAL.

5.5.4.66 int sg_httpsrv_set_thr_pool_size (struct sg_httpsrv * *srv*, unsigned int *size*)

Sets the size for the thread pool.

Parameters

in	*srv*	Server handle.
in	*size*	Thread pool size. Size greater than 1 enables the thread pooling.

Return values

0	Success.
EINVAL	Invalid argument.

Examples:

example_httpsrv_benchmark.c.

5.5.4.67 unsigned int sg_httpsrv_thr_pool_size (struct sg_httpsrv * *srv*)

Gets the size of the thread pool.

Parameters

in	*srv*	Server handle.

Returns

Thread pool size.

Return values

0	If the **srv** is null and sets the errno to EINVAL.

5.5.4.68 int sg_httpsrv_set_con_timeout (struct sg_httpsrv * *srv*, unsigned int *timeout*)

Sets the inactivity time to a client get time out.

Parameters

in	*srv*	Server handle.
in	*timeout*	Timeout (in seconds). Use zero for infinity timeout.

Return values

0	Success.
EINVAL	Invalid argument.

5.5.4.69 unsigned int sg_httpsrv_con_timeout (struct sg_httpsrv * *srv*)

Gets the inactivity time to a client get time out.

Parameters

in	*srv*	Server handle.

Returns

Timeout (in seconds).

Return values

0	If the **srv** is null and sets the `errno` to `EINVAL`.

5.5.4.70 int sg_httpsrv_set_con_limit (struct sg_httpsrv * *srv*, unsigned int *limit*)

Sets the limit of concurrent connections.

Parameters

in	*srv*	Server handle.
in	*limit*	Concurrent connections limit. Use zero for no limit.

Return values

0	Success.
EINVAL	Invalid argument.

Examples:

example_httpsrv_benchmark.c.

5.5.4.71 unsigned int sg_httpsrv_con_limit (struct sg_httpsrv * *srv*)

Gets the limit of concurrent connections.

Parameters

in	*srv*	Server handle.

Returns

Concurrent connections limit.

Return values

0	If the **srv** is null and sets the `errno` to `EINVAL`.

5.6 Path routing

Data Structures

- struct sg_entrypoint
- struct sg_entrypoints
- struct sg_route
- struct sg_router

Typedefs

- typedef int(* sg_entrypoints_iter_cb)(void *cls, struct sg_entrypoint *entrypoint)
- typedef int(* sg_segments_iter_cb)(void *cls, unsigned int index, const char *segment)
- typedef int(* sg_vars_iter_cb)(void *cls, const char *name, const char *val)
- typedef void(* sg_route_cb)(void *cls, struct sg_route *route)
- typedef int(* sg_routes_iter_cb)(void *cls, struct sg_route *route)
- typedef int(* sg_router_dispatch_cb)(void *cls, const char *path, struct sg_route *route)
- typedef int(* sg_router_match_cb)(void *cls, struct sg_route *route)

Functions

- const char * sg_entrypoint_name (struct sg_entrypoint *entrypoint)
- int sg_entrypoint_set_user_data (struct sg_entrypoint *entrypoint, void *data)
- void * sg_entrypoint_user_data (struct sg_entrypoint *entrypoint)
- struct sg_entrypoints * sg_entrypoints_new (void) __attribute__((malloc))
- void sg_entrypoints_free (struct sg_entrypoints *entrypoints)
- int sg_entrypoints_add (struct sg_entrypoints *entrypoints, const char *path, void *user_data)
- int sg_entrypoints_rm (struct sg_entrypoints *entrypoints, const char *path)
- int sg_entrypoints_iter (struct sg_entrypoints *entrypoints, sg_entrypoints_iter_cb cb, void *cls)
- int sg_entrypoints_clear (struct sg_entrypoints *entrypoints)
- int sg_entrypoints_find (struct sg_entrypoints *entrypoints, struct sg_entrypoint **entrypoint, const char *path)
- void * sg_route_handle (struct sg_route *route)
- void * sg_route_match (struct sg_route *route)
- const char * sg_route_rawpattern (struct sg_route *route)
- char * sg_route_pattern (struct sg_route *route) __attribute__((malloc))
- const char * sg_route_path (struct sg_route *route)
- int sg_route_segments_iter (struct sg_route *route, sg_segments_iter_cb cb, void *cls)
- int sg_route_vars_iter (struct sg_route *route, sg_vars_iter_cb cb, void *cls)
- void * sg_route_user_data (struct sg_route *route)
- int sg_routes_add2 (struct sg_route **routes, struct sg_route **route, const char *pattern, char *errmsg, size_t errlen, sg_route_cb cb, void *cls)
- bool sg_routes_add (struct sg_route **routes, const char *pattern, sg_route_cb cb, void *cls)
- int sg_routes_rm (struct sg_route **routes, const char *pattern)
- int sg_routes_iter (struct sg_route *routes, sg_routes_iter_cb cb, void *cls)
- int sg_routes_next (struct sg_route **route)
- unsigned int sg_routes_count (struct sg_route *routes)
- int sg_routes_cleanup (struct sg_route **routes)
- struct sg_router * sg_router_new (struct sg_route *routes) __attribute__((malloc))
- void sg_router_free (struct sg_router *router)
- int sg_router_dispatch2 (struct sg_router *router, const char *path, void *user_data, sg_router_dispatch_cb dispatch_cb, void *cls, sg_router_match_cb match_cb)
- int sg_router_dispatch (struct sg_router *router, const char *path, void *user_data)

5.6.1 Detailed Description

High-performance path routing.

5.6.2 Typedef Documentation

5.6.2.1 typedef int(∗ sg_entrypoints_iter_cb)(void ∗cls, struct sg_entrypoint ∗entrypoint)

Callback signature used by sg_entrypoints_iter() to iterate entry-point items.

Parameters

out	*cls*	User-defined closure.
out	*pair*	Current iterated entry-point.

Return values

0	Success.
E<ERROR>	User-defined error to stop the items iteration.

5.6.2.2 typedef int(∗ sg_segments_iter_cb)(void ∗cls, unsigned int index, const char ∗segment)

Callback signature used by sg_route_segments_iter() to iterate the path segments.

Parameters

out	*cls*	User-defined closure.
out	*index*	Current iterated item index.
out	*segment*	Current iterated segment.

Return values

0	Success.
E<ERROR>	User-defined error to stop the segments iteration.

5.6.2.3 typedef int(∗ sg_vars_iter_cb)(void ∗cls, const char ∗name, const char ∗val)

Callback signature used by sg_route_vars_iter() to iterate the path variables.

Parameters

out	*cls*	User-defined closure.
out	*name*	Current iterated variable name.
out	*val*	Current iterated variable value.

Return values

0	Success.
E<ERROR>	User-defined error to stop the variables iteration.

5.6.2.4 typedef void(∗ sg_route_cb)(void ∗cls, struct sg_route ∗route)

Callback signature used to handle the path routing.

Parameters

out	*cls*	User-defined closure.

out	*route*	Route handle.

5.6.2.5 typedef int(∗ sg_routes_iter_cb)(void ∗cls, struct sg_route ∗route)

Callback signature used by sg_routes_iter() to iterate route items.

Parameters

out	*cls*	User-defined closure.
out	*route*	Current iterated route.

Return values

0	Success.
E<ERROR>	User-defined error to stop the route items iteration.

5.6.2.6 typedef int(∗ sg_router_dispatch_cb)(void ∗cls, const char ∗path, struct sg_route ∗route)

Callback signature used by sg_router_dispatch2 in the route dispatching loop.

Parameters

out	*cls*	User-defined closure.
out	*path*	Route path as null-terminated string.
out	*route*	Route handle.

Return values

0	Success.
E<ERROR>	User-defined error to stop the route dispatching loop.

5.6.2.7 typedef int(∗ sg_router_match_cb)(void ∗cls, struct sg_route ∗route)

Callback signature used by sg_router_dispatch2 when the path matches the pattern before the route dispatching.

Parameters

out	*cls*	User-defined closure.
out	*route*	Route handle.

Return values

0	Success.
E<ERROR>	User-defined error to stop the route dispatching.

5.6.3 Function Documentation

5.6.3.1 const char∗ sg_entrypoint_name (struct sg_entrypoint ∗ *entrypoint*)

Returns the name of the entry-point handle **entrypoint**.

Parameters

in	*entrypoint*	Entry-point handle.

Returns

Entry-point name as null-terminated string.

Return values

NULL	If the **entrypoint** is null and sets the errno to EINVAL.

5.6.3.2 int sg_entrypoint_set_user_data (struct sg_entrypoint * *entrypoint*, void * *data*)

Sets user data to the entry-point handle.

Parameters

in	*entrypoint*	Entry-point handle.
in	*data*	User data pointer.

Return values

0	Success.
EINVAL	Invalid argument.

5.6.3.3 void* sg_entrypoint_user_data (struct sg_entrypoint * *entrypoint*)

Gets user data from the entry-point handle.

Parameters

in	*entrypoint*	Entry-point handle.

Returns

User data pointer.

Return values

NULL	If **entrypoint** is null and sets the errno to EINVAL.

Examples:

example_entrypoint.c.

5.6.3.4 struct sg_entrypoints* sg_entrypoints_new (void)

Creates a new entry-points handle.

Returns

Entry-points handle.

Return values

NULL	If no memory space is available.

Examples:

example_entrypoint.c.

5.6.3.5 void sg_entrypoints_free (struct sg_entrypoints * *entrypoints*)

Frees the entry-points handle previously allocated by sg_entrypoints_new().

Parameters

in	*entrypoints*	Pointer of the entry-points to be freed.

Examples:

example_entrypoint.c.

5.6.3.6 int sg_entrypoints_add (struct sg_entrypoints * *entrypoints*, const char * *path*, void * *user_data*)

Adds a new entry-point item to the entry-points **entrypoints**.

Parameters

in	*entrypoints*	Entry-points handle.
in	*path*	Entry-point path.
in	*user_data*	User data pointer.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.
EALREADY	Entry-point already added.

Examples:

example_entrypoint.c.

5.6.3.7 int sg_entrypoints_rm (struct sg_entrypoints * *entrypoints*, const char * *path*)

Removes an entry-point item from the entry-points **entrypoints**.

Parameters

in	*entrypoints*	Entry-points handle.
in	*path*	Entry-point path to be removed.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.
ENOENT	Entry-point already removed.

5.6.3.8 int sg_entrypoints_iter (struct sg_entrypoints * *entrypoints*, sg_entrypoints_iter_cb *cb*, void * *cls*)

Iterates over entry-point items.

Parameters

in	*entrypoints*	Entry-points handle.
in	*cb*	Callback to iterate the entry-point items.
in,out	*cls*	User-specified value.

Return values

0	Success.
EINVAL	Invalid argument.

Returns

Callback result when it is different from `0`.

5.6.3.9 int sg_entrypoints_clear (struct sg_entrypoints * *entrypoints*)

Clears all existing entry-point items in the entry-points **`entrypoints`**.

Parameters

`in`	*entrypoints*	Entry-points handle.

Return values

0	Success.
EINVAL	Invalid argument.

5.6.3.10 int sg_entrypoints_find (struct sg_entrypoints * *entrypoints*, struct sg_entrypoint ** *entrypoint*, const char * *path*)

Finds an entry-point item by path.

Parameters

`in`	*entrypoints*	Entry-points handle.
`in,out`	*entrypoint*	Pointer of the variable to store the found entry-point.
`in`	*path*	Entry-point path to be found.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.
ENOENT	Pair not found.

Examples:

example_entrypoint.c.

5.6.3.11 void* sg_route_handle (struct sg_route * *route*)

Returns the PCRE2 handle containing the compiled regex code.

Parameters

`in`	*route*	Route handle.

Returns

PCRE2 handle containing the compiled regex code.

Return values

NULL	If **route** is null and sets the errno to EINVAL.

5.6.3.12 void* sg_route_match (struct sg_route * *route*)

Returns the PCRE2 match data created from the route pattern.

Parameters

in	*route*	Route handle.

Returns

PCRE2 match data.

Return values

NULL	If **route** is null and sets the errno to EINVAL.

5.6.3.13 const char* sg_route_rawpattern (struct sg_route * *route*)

Returns the raw route pattern. For example, given a pattern /foo, the raw pattern is ^/foo$.

Parameters

in	*route*	Route handle.

Returns

Raw pattern as null-terminated string.

Return values

NULL	If **route** is null and sets the errno to EINVAL.

5.6.3.14 char* sg_route_pattern (struct sg_route * *route*)

Returns the route pattern.

Parameters

in	*route*	Route handle.

Returns

Pattern as null-terminated string.

Return values

NULL	If **route** is null and sets the errno to EINVAL.
NULL	If no memory space is available and sets the errno to ENOMEM.

Warning

The caller must free the returned value.

5.6.3.15 const char* sg_route_path (struct sg_route * *route*)

Returns the route path.

Parameters

in	*route*	Route handle.

Returns

Path component as null-terminated string.

Return values

NULL	If **route** is null and sets the errno to EINVAL.

Examples:

example_entrypoint.c, example_router_segments.c, example_router_simple.c, and example_router_vars.c.

5.6.3.16 int sg_route_segments_iter (struct sg_route * *route*, sg_segments_iter_cb *cb*, void * *cls*)

Iterates over path segments.

Parameters

in	*route*	Route handle.
in	*cb*	Callback to iterate the path segments.
in,out	*cls*	User-specified value.

Return values

0	Success.
EINVAL	Invalid argument.

Returns

Callback result when it is different from 0.

Examples:

example_router_segments.c.

5.6.3.17 int sg_route_vars_iter (struct sg_route * *route*, sg_vars_iter_cb *cb*, void * *cls*)

Iterates over path variables.

Parameters

in	*route*	Route handle.
in	*cb*	Callback to iterate the path variables.
in,out	*cls*	User-specified value.

Return values

0	Success.
EINVAL	Invalid argument.
ENOMEM	Out of memory.

Returns

Callback result when it is different from 0.

Examples:

example_router_srv.c, and example_router_vars.c.

5.6.3.18 void* sg_route_user_data (struct sg_route * *route*)

Gets user data from the route handle.

Parameters

in	*route*	Route handle.

Returns

User data pointer.

Return values

NULL	If **route** is null and sets the errno to EINVAL.

Examples:

example_router_srv.c.

5.6.3.19 int sg_routes_add2 (struct sg_route ** *routes*, struct sg_route ** *route*, const char * *pattern*, char * *errmsg*, size_t *errlen*, sg_route_cb *cb*, void * *cls*)

Adds a route item to the route list **routes**.

Parameters

in,out	*routes*	Route list pointer to add a new route item.
in,out	*route*	Pointer of the variable to store the added route reference.
in	*pattern*	Pattern as null-terminated string. It must be a valid regular expression in PCRE2 syntax.
in,out	*errmsg*	Pointer of a string to store the error message.
in	*errlen*	Length of the error message.
in	*cb*	Callback to handle the path routing.
in	*cls*	User-defined closure.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Route already added.
ENOMEM	Out of memory.

Note

The pattern is enclosed between ^ and $ automatically if it does not start with (.
The escape sequence \K is not supported. It causes EINVAL if used.
The pattern is compiled using just-in-time optimization (JIT) when it is supported.

5.6.3.20 bool sg_routes_add (struct sg_route ** *routes*, const char * *pattern*, sg_route_cb *cb*, void * *cls*)

Adds a route item to the route list **routes**. It uses the stderr to print the validation errors.

Parameters

in,out	*routes*	Route list pointer to add a new route item.
in	*pattern*	Pattern as null-terminated string. It must be a valid regular expression in PCRE2 syntax.
in	*cb*	Callback to handle the path routing.

in	*cls*	User-defined closure.

Return values

0	Success.
EINVAL	Invalid argument.
EALREADY	Route already added.
ENOMEM	Out of memory.

Note

The pattern is enclosed between ^ and $ automatically if it does not start with (.
The escape sequence \K is not supported. It causes EINVAL if used.
The pattern is compiled using just-in-time optimization (JIT) when it is supported.

Examples:

example_entrypoint.c, example_router_segments.c, example_router_simple.c, example_router_srv.c, and example_router_vars.c.

5.6.3.21 int sg_routes_rm (struct sg_route ** *routes*, const char * *pattern*)

Removes a route item from the route list **routes**.

Parameters

in,out	*routes*	Route list pointer to add a new route item.
in	*pattern*	Pattern as null-terminated string of the route to be removed.

Return values

0	Success.
EINVAL	Invalid argument.
ENOENT	Route already removed.

5.6.3.22 int sg_routes_iter (struct sg_route * *routes*, sg_routes_iter_cb *cb*, void * *cls*)

Iterates over all the routes in the route list **routes**.

Parameters

in	*routes*	Route list handle.
in	*cb*	Callback to iterate over route items.
in	*cls*	User-defined closure.

Return values

0	Success.
EINVAL	Invalid argument.
E<ERROR>	User-defined error to abort the list iteration.

5.6.3.23 int sg_routes_next (struct sg_route ** *route*)

Returns the next route int the route list.

Parameters

in,out	*route*	Pointer to the next route item.

Return values

0	Success.
EINVAL	Invalid argument.

5.6.3.24 unsigned int sg_routes_count (struct sg_route * *routes*)

Counts the total routes in the route list **routes**.

Parameters

in	*routes*	Route list handle.

Returns

Total of route items.

Return values

0	If the list is empty or null.

5.6.3.25 int sg_routes_cleanup (struct sg_route ** *routes*)

Cleans the entire route list.

Parameters

in,out	*routes*	Pointer to the route list.

Examples:

example_entrypoint.c, example_router_segments.c, example_router_simple.c, example_router_srv.c, and example_router_vars.c.

5.6.3.26 struct sg_router* sg_router_new (struct sg_route * *routes*)

Creates a new path router handle. It requires a filled route list **routes**.

Parameters

in	*routes*	Route list handle.

Returns

New router handle.

Return values

NULL	If the **routes** is null and sets the errno to EINVAL or no memory space.

Examples:

example_entrypoint.c, example_router_segments.c, example_router_simple.c, example_router_srv.c, and example_router_vars.c.

5.6.3.27 void sg_router_free (struct sg_router * *router*)

Frees the router handle previously allocated by sg_router_new().

Parameters

in	*router*	Router handle.

Examples:

example_entrypoint.c, example_router_segments.c, example_router_simple.c, example_router_srv.c, and example_router_vars.c.

5.6.3.28 int sg_router_dispatch2 (struct sg_router * *router*, const char * *path*, void * *user_data*, sg_router_dispatch_cb *dispatch_cb*, void * *cls*, sg_router_match_cb *match_cb*)

Dispatches a route that its pattern matches the path passed in **path**.

Parameters

in	*router*	Router handle.
in	*path*	Path to dispatch a route.
in	*user_data*	User data pointer to be hold by the route.
in	*dispatch_cb*	Callback triggered for each route item in the route dispatching loop.
in	*cls*	User-defined closure passed to the **dispatch_cb** and **match_cb** callbacks.
in	*match_cb*	Callback triggered when the path matches the route pattern.

Return values

0	Success.
EINVAL	Invalid argument.
ENOENT	Route not found or path not matched.
E<ERROR>	User-defined error in **dispatch_cb** or **match_cb**.

Note

The route callback sg_route_cb is triggered when the path matches the route pattern.
The match logic uses just-in-time optimization (JIT) when it is supported.

5.6.3.29 int sg_router_dispatch (struct sg_router * *router*, const char * *path*, void * *user_data*)

Dispatches a route that its pattern matches the path passed in **path**.

Parameters

in	*router*	Router handle.
in	*path*	Path to dispatch a route.
in	*user_data*	User data pointer to be hold by the route.

Return values

0	Success.
EINVAL	Invalid argument.
ENOENT	Route not found or path not matched.
E<ERROR>	User-defined error in **dispatch_cb** or **match_cb**.

Note

The route callback sg_route_cb is triggered when the path matches the route pattern.
The match logic uses just-in-time optimization (JIT) when it is supported.

Examples:

example_entrypoint.c, example_router_segments.c, example_router_simple.c, example_router_srv.c, and example_router_vars.c.

6 Data Structure Documentation

6.1 sg_entrypoint Struct Reference

`#include <sagui.h>`

6.1.1 Detailed Description

Handle for the entry-point handling. It defines an entry-point to a path or resource. For example, given a path `/api1/customer`, the part considered as entry-point is `/api1`.

Examples:

example_entrypoint.c.

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

- sagui.h

6.2 sg_entrypoints Struct Reference

`#include <sagui.h>`

6.2.1 Detailed Description

Handle for the entry-point list. It is used to create a list of entry-point sg_entrypoint.

Examples:

example_entrypoint.c.

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

- sagui.h

6.3 sg_httpauth Struct Reference

`#include <sagui.h>`

6.3.1 Detailed Description

Handle for the HTTP basic authentication.

Examples:

example_httpauth.c.

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

- sagui.h

6.4 sg_httpreq Struct Reference

`#include <sagui.h>`

6.4.1 Detailed Description

Handle for the request handling. It contains headers, cookies, query-string, fields, payloads, uploads and other data sent by the client.

Examples:

example_httpauth.c, example_httpcomp.c, example_httpcookie.c, example_httpreq_payload.c, example_↩ httpsrv.c, example_httpsrv_benchmark.c, example_httpsrv_tls.c, example_httpsrv_tls_cert_auth.c, example↩ _httpuplds.c, and example_router_srv.c.

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

- sagui.h

6.5 sg_httpres Struct Reference

`#include <sagui.h>`

6.5.1 Detailed Description

Handle for the response handling. It dispatches headers, contents, binaries, files and other data to the client.

Examples:

example_httpauth.c, example_httpcomp.c, example_httpcookie.c, example_httpreq_payload.c, example_↩ httpsrv.c, example_httpsrv_benchmark.c, example_httpsrv_tls.c, example_httpsrv_tls_cert_auth.c, example↩ _httpuplds.c, and example_router_srv.c.

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

- sagui.h

6.6 sg_httpsrv Struct Reference

`#include <sagui.h>`

6.6.1 Detailed Description

Handle for the fast event-driven HTTP server.

Examples:

example_httpauth.c, example_httpcomp.c, example_httpcookie.c, example_httpreq_payload.c, example_↩ httpsrv.c, example_httpsrv_benchmark.c, example_httpsrv_tls.c, example_httpsrv_tls_cert_auth.c, example↩ _httpuplds.c, and example_router_srv.c.

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

- sagui.h

6.7 sg_httpupld Struct Reference

`#include <sagui.h>`

6.7.1 Detailed Description

Handle for the upload handling. It is used to represent a single upload or a list of uploads.

Examples:

example_httpuplds.c.

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

- sagui.h

6.8 sg_route Struct Reference

`#include <sagui.h>`

6.8.1 Detailed Description

Handle for the route item. It holds a user data to be dispatched when a path matches the user defined pattern (route pattern).

Examples:

example_entrypoint.c, example_router_segments.c, example_router_simple.c, example_router_srv.c, and example_router_vars.c.

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

- sagui.h

6.9 sg_router Struct Reference

`#include <sagui.h>`

6.9.1 Detailed Description

Handle for the path router. It holds the reference of a route list to be dispatched.

Examples:

example_entrypoint.c, example_router_segments.c, example_router_simple.c, example_router_srv.c, and example_router_vars.c.

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

- sagui.h

6.10 sg_str Struct Reference

`#include <sagui.h>`

6.10.1 Detailed Description

Handle for the string structure used to represent a HTML body, POST payload and more.

Examples:

example_httpreq_payload.c, example_httpuplds.c, example_router_srv.c, and example_str.c.

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

- sagui.h

6.11 sg_strmap Struct Reference

`#include <sagui.h>`

6.11.1 Detailed Description

Handle for hash table that maps name-value pairs. It is useful to represent posting fields, query-string parameter, client cookies and more.

Examples:

example_httpcomp.c, example_httpcookie.c, example_httpuplds.c, and example_strmap.c.

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

- sagui.h

7 File Documentation

7.1 example_entrypoint.h File Reference

7.2 example_httpauth.h File Reference

7.3 example_httpcomp.h File Reference

7.4 example_httpcookie.h File Reference

7.5 example_httpreq_payload.h File Reference

7.6 example_httpsrv.h File Reference

7.7 example_httpsrv_benchmark.h File Reference

7.8 example_httpsrv_tls.h File Reference

7.9 example_httpsrv_tls_cert_auth.h File Reference

7.10 example_httpuplds.h File Reference

7.11 example_router_segments.h File Reference

7.12 example_router_simple.h File Reference

7.13 example_router_srv.h File Reference

7.14 example_router_vars.h File Reference

7.15 example_str.h File Reference

7.16 example_strmap.h File Reference

7.17 sagui.h File Reference

```
#include <stdio.h>
#include <stddef.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdarg.h>
#include <string.h>
#include <time.h>
```

Macros

- #define SG_ERR_SIZE 256
- #define sg_httpres_send(res, val, content_type, status)
- #define sg_httpres_download(res, filename) sg_httpres_sendfile2((res), 0, 0, 0, (filename), "attachment", 200)
- #define sg_httpres_render(res, filename) sg_httpres_sendfile2((res), 0, 0, 0, (filename), "inline", 200)
- #define sg_httpres_zsend(res, val, content_type, status)
- #define sg_httpres_zdownload(res, filename) sg_httpres_zsendfile2((res), 1, 0, 0, 0, (filename), "attachment", 200)
- #define sg_httpres_zrender(res, filename) sg_httpres_zsendfile2((res), 1, 0, 0, 0, (filename), "inline", 200)

Typedefs

- typedef void *(* sg_malloc_func)(size_t size)
- typedef void *(* sg_realloc_func)(void *ptr, size_t size)
- typedef void(* sg_free_func)(void *ptr)
- typedef void(* sg_err_cb)(void *cls, const char *err)
- typedef ssize_t(* sg_write_cb)(void *handle, uint64_t offset, const char *buf, size_t size)
- typedef ssize_t(* sg_read_cb)(void *handle, uint64_t offset, char *buf, size_t size)
- typedef void(* sg_free_cb)(void *handle)
- typedef int(* sg_save_cb)(void *handle, bool overwritten)
- typedef int(* sg_save_as_cb)(void *handle, const char *path, bool overwritten)
- typedef int(* sg_strmap_iter_cb)(void *cls, struct sg_strmap *pair)
- typedef int(* sg_strmap_sort_cb)(void *cls, struct sg_strmap *pair_a, struct sg_strmap *pair_b)
- typedef void(* sg_httpsrv_cli_cb)(void *cls, const void *client, bool *closed)
- typedef bool(* sg_httpauth_cb)(void *cls, struct sg_httpauth *auth, struct sg_httpreq *req, struct sg_httpres *res)
- typedef int(* sg_httpupld_cb)(void *cls, void **handle, const char *dir, const char *field, const char *name, const char *mime, const char *encoding)
- typedef int(* sg_httpuplds_iter_cb)(void *cls, struct sg_httpupld *upld)
- typedef void(* sg_httpreq_cb)(void *cls, struct sg_httpreq *req, struct sg_httpres *res)
- typedef int(* sg_entrypoints_iter_cb)(void *cls, struct sg_entrypoint *entrypoint)

- typedef int(* sg_segments_iter_cb)(void *cls, unsigned int index, const char *segment)
- typedef int(* sg_vars_iter_cb)(void *cls, const char *name, const char *val)
- typedef void(* sg_route_cb)(void *cls, struct sg_route *route)
- typedef int(* sg_routes_iter_cb)(void *cls, struct sg_route *route)
- typedef int(* sg_router_dispatch_cb)(void *cls, const char *path, struct sg_route *route)
- typedef int(* sg_router_match_cb)(void *cls, struct sg_route *route)

Functions

- unsigned int sg_version (void)
- const char * sg_version_str (void)
- int sg_mm_set (sg_malloc_func malloc_func, sg_realloc_func realloc_func, sg_free_func free_func)
- void * sg_malloc (size_t size) __attribute__((malloc))
- void * sg_alloc (size_t size) __attribute__((malloc))
- void * sg_realloc (void *ptr, size_t size) __attribute__((malloc))
- void sg_free (void *ptr)
- char * sg_strerror (int errnum, char *errmsg, size_t errlen)
- bool sg_is_post (const char *method)
- char * sg_extract_entrypoint (const char *path)
- char * sg_tmpdir (void)
- ssize_t sg_eor (bool err)
- int sg_ip (const void *socket, char *buf, size_t size)
- struct sg_str * sg_str_new (void) __attribute__((malloc))
- void sg_str_free (struct sg_str *str)
- int sg_str_write (struct sg_str *str, const char *val, size_t len)
- int sg_str_printf_va (struct sg_str *str, const char *fmt, va_list ap)
- int sg_str_printf (struct sg_str *str, const char *fmt,...) __attribute__((format(printf
- int const char * sg_str_content (struct sg_str *str)
- size_t sg_str_length (struct sg_str *str)
- int sg_str_clear (struct sg_str *str)
- const char * sg_strmap_name (struct sg_strmap *pair)
- const char * sg_strmap_val (struct sg_strmap *pair)
- int sg_strmap_add (struct sg_strmap **map, const char *name, const char *val)
- int sg_strmap_set (struct sg_strmap **map, const char *name, const char *val)
- int sg_strmap_find (struct sg_strmap *map, const char *name, struct sg_strmap **pair)
- const char * sg_strmap_get (struct sg_strmap *map, const char *name)
- int sg_strmap_rm (struct sg_strmap **map, const char *name)
- int sg_strmap_iter (struct sg_strmap *map, sg_strmap_iter_cb cb, void *cls)
- int sg_strmap_sort (struct sg_strmap **map, sg_strmap_sort_cb cb, void *cls)
- unsigned int sg_strmap_count (struct sg_strmap *map)
- int sg_strmap_next (struct sg_strmap **next)
- void sg_strmap_cleanup (struct sg_strmap **map)
- int sg_httpauth_set_realm (struct sg_httpauth *auth, const char *realm)
- const char * sg_httpauth_realm (struct sg_httpauth *auth)
- int sg_httpauth_deny2 (struct sg_httpauth *auth, const char *reason, const char *content_type, unsigned int status)
- int sg_httpauth_deny (struct sg_httpauth *auth, const char *reason, const char *content_type)
- int sg_httpauth_cancel (struct sg_httpauth *auth)
- const char * sg_httpauth_usr (struct sg_httpauth *auth)
- const char * sg_httpauth_pwd (struct sg_httpauth *auth)
- int sg_httpuplds_iter (struct sg_httpupld *uplds, sg_httpuplds_iter_cb cb, void *cls)
- int sg_httpuplds_next (struct sg_httpupld **upld)
- unsigned int sg_httpuplds_count (struct sg_httpupld *uplds)
- void * sg_httpupld_handle (struct sg_httpupld *upld)

- const char * sg_httpupld_dir (struct sg_httpupld *upld)
- const char * sg_httpupld_field (struct sg_httpupld *upld)
- const char * sg_httpupld_name (struct sg_httpupld *upld)
- const char * sg_httpupld_mime (struct sg_httpupld *upld)
- const char * sg_httpupld_encoding (struct sg_httpupld *upld)
- uint64_t sg_httpupld_size (struct sg_httpupld *upld)
- int sg_httpupld_save (struct sg_httpupld *upld, bool overwritten)
- int sg_httpupld_save_as (struct sg_httpupld *upld, const char *path, bool overwritten)
- struct sg_strmap ** sg_httpreq_headers (struct sg_httpreq *req)
- struct sg_strmap ** sg_httpreq_cookies (struct sg_httpreq *req)
- struct sg_strmap ** sg_httpreq_params (struct sg_httpreq *req)
- struct sg_strmap ** sg_httpreq_fields (struct sg_httpreq *req)
- const char * sg_httpreq_version (struct sg_httpreq *req)
- const char * sg_httpreq_method (struct sg_httpreq *req)
- const char * sg_httpreq_path (struct sg_httpreq *req)
- struct sg_str * sg_httpreq_payload (struct sg_httpreq *req)
- bool sg_httpreq_is_uploading (struct sg_httpreq *req)
- struct sg_httpupld * sg_httpreq_uploads (struct sg_httpreq *req)
- const void * sg_httpreq_client (struct sg_httpreq *req)
- void * sg_httpreq_tls_session (struct sg_httpreq *req)
- int sg_httpreq_set_user_data (struct sg_httpreq *req, void *data)
- void * sg_httpreq_user_data (struct sg_httpreq *req)
- struct sg_strmap ** sg_httpres_headers (struct sg_httpres *res)
- int sg_httpres_set_cookie (struct sg_httpres *res, const char *name, const char *val)
- int sg_httpres_sendbinary (struct sg_httpres *res, void *buf, size_t size, const char *content_type, unsigned int status)
- int sg_httpres_sendfile2 (struct sg_httpres *res, uint64_t size, uint64_t max_size, uint64_t offset, const char *filename, const char *disposition, unsigned int status)
- int sg_httpres_sendfile (struct sg_httpres *res, uint64_t size, uint64_t max_size, uint64_t offset, const char *filename, bool downloaded, unsigned int status)
- int sg_httpres_sendstream (struct sg_httpres *res, uint64_t size, sg_read_cb read_cb, void *handle, sg_free_cb free_cb, unsigned int status)
- int sg_httpres_zsendbinary2 (struct sg_httpres *res, int level, void *buf, size_t size, const char *content_type, unsigned int status)
- int sg_httpres_zsendbinary (struct sg_httpres *res, void *buf, size_t size, const char *content_type, unsigned int status)
- int sg_httpres_zsendstream2 (struct sg_httpres *res, int level, uint64_t size, sg_read_cb read_cb, void *handle, sg_free_cb free_cb, unsigned int status)
- int sg_httpres_zsendstream (struct sg_httpres *res, sg_read_cb read_cb, void *handle, sg_free_cb free_cb, unsigned int status)
- int sg_httpres_zsendfile2 (struct sg_httpres *res, int level, uint64_t size, uint64_t max_size, uint64_t offset, const char *filename, const char *disposition, unsigned int status)
- int sg_httpres_zsendfile (struct sg_httpres *res, uint64_t size, uint64_t max_size, uint64_t offset, const char *filename, bool downloaded, unsigned int status)
- int sg_httpres_clear (struct sg_httpres *res)
- struct sg_httpsrv * sg_httpsrv_new2 (sg_httpauth_cb auth_cb, sg_httpreq_cb req_cb, sg_err_cb err_cb, void *cls) __attribute__((malloc))
- struct sg_httpsrv * sg_httpsrv_new (sg_httpreq_cb cb, void *cls) __attribute__((malloc))
- void sg_httpsrv_free (struct sg_httpsrv *srv)
- bool sg_httpsrv_tls_listen2 (struct sg_httpsrv *srv, const char *key, const char *pwd, const char *cert, const char *trust, const char *dhparams, uint16_t port, bool threaded)
- bool sg_httpsrv_tls_listen (struct sg_httpsrv *srv, const char *key, const char *cert, uint16_t port, bool threaded)
- bool sg_httpsrv_listen (struct sg_httpsrv *srv, uint16_t port, bool threaded)
- int sg_httpsrv_shutdown (struct sg_httpsrv *srv)

- uint16_t sg_httpsrv_port (struct sg_httpsrv *srv)
- bool sg_httpsrv_is_threaded (struct sg_httpsrv *srv)
- int sg_httpsrv_set_cli_cb (struct sg_httpsrv *srv, sg_httpsrv_cli_cb cb, void *cls)
- int sg_httpsrv_set_upld_cbs (struct sg_httpsrv *srv, sg_httpupld_cb cb, void *cls, sg_write_cb write_cb, sg_free_cb free_cb, sg_save_cb save_cb, sg_save_as_cb save_as_cb)
- int sg_httpsrv_set_upld_dir (struct sg_httpsrv *srv, const char *dir)
- const char * sg_httpsrv_upld_dir (struct sg_httpsrv *srv)
- int sg_httpsrv_set_post_buf_size (struct sg_httpsrv *srv, size_t size)
- size_t sg_httpsrv_post_buf_size (struct sg_httpsrv *srv)
- int sg_httpsrv_set_payld_limit (struct sg_httpsrv *srv, size_t limit)
- size_t sg_httpsrv_payld_limit (struct sg_httpsrv *srv)
- int sg_httpsrv_set_uplds_limit (struct sg_httpsrv *srv, uint64_t limit)
- uint64_t sg_httpsrv_uplds_limit (struct sg_httpsrv *srv)
- int sg_httpsrv_set_thr_pool_size (struct sg_httpsrv *srv, unsigned int size)
- unsigned int sg_httpsrv_thr_pool_size (struct sg_httpsrv *srv)
- int sg_httpsrv_set_con_timeout (struct sg_httpsrv *srv, unsigned int timeout)
- unsigned int sg_httpsrv_con_timeout (struct sg_httpsrv *srv)
- int sg_httpsrv_set_con_limit (struct sg_httpsrv *srv, unsigned int limit)
- unsigned int sg_httpsrv_con_limit (struct sg_httpsrv *srv)
- const char * sg_entrypoint_name (struct sg_entrypoint *entrypoint)
- int sg_entrypoint_set_user_data (struct sg_entrypoint *entrypoint, void *data)
- void * sg_entrypoint_user_data (struct sg_entrypoint *entrypoint)
- struct sg_entrypoints * sg_entrypoints_new (void) __attribute__((malloc))
- void sg_entrypoints_free (struct sg_entrypoints *entrypoints)
- int sg_entrypoints_add (struct sg_entrypoints *entrypoints, const char *path, void *user_data)
- int sg_entrypoints_rm (struct sg_entrypoints *entrypoints, const char *path)
- int sg_entrypoints_iter (struct sg_entrypoints *entrypoints, sg_entrypoints_iter_cb cb, void *cls)
- int sg_entrypoints_clear (struct sg_entrypoints *entrypoints)
- int sg_entrypoints_find (struct sg_entrypoints *entrypoints, struct sg_entrypoint **entrypoint, const char *path)
- void * sg_route_handle (struct sg_route *route)
- void * sg_route_match (struct sg_route *route)
- const char * sg_route_rawpattern (struct sg_route *route)
- char * sg_route_pattern (struct sg_route *route) __attribute__((malloc))
- const char * sg_route_path (struct sg_route *route)
- int sg_route_segments_iter (struct sg_route *route, sg_segments_iter_cb cb, void *cls)
- int sg_route_vars_iter (struct sg_route *route, sg_vars_iter_cb cb, void *cls)
- void * sg_route_user_data (struct sg_route *route)
- int sg_routes_add2 (struct sg_route **routes, struct sg_route **route, const char *pattern, char *errmsg, size_t errlen, sg_route_cb cb, void *cls)
- bool sg_routes_add (struct sg_route **routes, const char *pattern, sg_route_cb cb, void *cls)
- int sg_routes_rm (struct sg_route **routes, const char *pattern)
- int sg_routes_iter (struct sg_route *routes, sg_routes_iter_cb cb, void *cls)
- int sg_routes_next (struct sg_route **route)
- unsigned int sg_routes_count (struct sg_route *routes)
- int sg_routes_cleanup (struct sg_route **routes)
- struct sg_router * sg_router_new (struct sg_route *routes) __attribute__((malloc))
- void sg_router_free (struct sg_router *router)
- int sg_router_dispatch2 (struct sg_router *router, const char *path, void *user_data, sg_router_dispatch_cb dispatch_cb, void *cls, sg_router_match_cb match_cb)
- int sg_router_dispatch (struct sg_router *router, const char *path, void *user_data)

7.17.1 Macro Definition Documentation

7.17.1.1 #define SG_ERR_SIZE 256

8 Example Documentation

8.1 example_entrypoint.c

Simple example showing the path entry-point feature.

```
/*
 *                          _
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

static void r1_route_cb(void *cls, struct sg_route *route) {
  fprintf(stdout, "%s: %s\n", sg_route_path(route), (const char *) cls);
  fflush(stdout);
}

static void r2_route_cb(void *cls, struct sg_route *route) {
  fprintf(stdout, "%s: %s\n", sg_route_path(route), (const char *) cls);
  fflush(stdout);
}

int main(void) {
  struct sg_router *r1, *r2;
  struct sg_route *rts1 = NULL, *rts2 = NULL;
  struct sg_entrypoints *entrypoints;
  struct sg_entrypoint *entrypoint;
  sg_routes_add(&rts1, "/foo", r1_route_cb, "r1-foo-data");
  sg_routes_add(&rts1, "/bar", r1_route_cb, "r1-bar-data");
  r1 = sg_router_new(rts1);
  sg_routes_add(&rts2, "/foo", r2_route_cb, "r2-foo-data");
  sg_routes_add(&rts2, "/bar", r2_route_cb, "r2-bar-data");
  r2 = sg_router_new(rts2);

  entrypoints = sg_entrypoints_new();
  sg_entrypoints_add(entrypoints, "/r1", r1);
  sg_entrypoints_add(entrypoints, "/r2", r2);

  sg_entrypoints_find(entrypoints, &entrypoint, "/r1/foo");
  sg_router_dispatch(sg_entrypoint_user_data(entrypoint), "/foo",
      NULL);
  sg_entrypoints_find(entrypoints, &entrypoint, "/r1/bar");
  sg_router_dispatch(sg_entrypoint_user_data(entrypoint), "/bar",
      NULL);
  sg_entrypoints_find(entrypoints, &entrypoint, "/r2/foo");
  sg_router_dispatch(sg_entrypoint_user_data(entrypoint), "/foo",
      NULL);
  sg_entrypoints_find(entrypoints, &entrypoint, "/r2/bar");
  sg_router_dispatch(sg_entrypoint_user_data(entrypoint), "/bar",
      NULL);
```

```
  sg_routes_cleanup(&rts1);
  sg_routes_cleanup(&rts2);
  sg_router_free(r1);
  sg_router_free(r2);
  sg_entrypoints_free(entrypoints);
  return EXIT_SUCCESS;
}
```

8.2 example_httpauth.c

Simple example showing the Basic authentication feature.

```
/*
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

static bool strmatch(const char *s1, const char *s2) {
  if (!s1 || !s2)
    return false;
  return strcmp(s1, s2) == 0;
}

static bool auth_cb(__SG_UNUSED void *cls, struct sg_httpauth *auth,
                    __SG_UNUSED struct sg_httpreq *req,
                    __SG_UNUSED struct sg_httpres *res) {
  bool pass;
  sg_httpauth_set_realm(auth, "My realm");
  pass = strmatch(sg_httpauth_usr(auth), "abc") &&
         strmatch(sg_httpauth_pwd(auth), "123");
  if (!pass)
    sg_httpauth_deny(auth,
                     "<html><head><title>Denied</title></head><body><font "
                     "color=\"red\">Go away</font></body></html>",
                     "text/html; charset=utf-8");
  return pass;
}

static void err_cb(__SG_UNUSED void *cls, const char *err) {
  fprintf(stderr, "%s", err);
  fflush(stderr);
}

static void req_cb(__SG_UNUSED void *cls, __SG_UNUSED struct sg_httpreq *req,
                   struct sg_httpres *res) {
  sg_httpres_send(res,
                  "<html><head><title>Secret</title></head><body><font "
                  "color=\"green\">Secret page</font></body></html>",
                  "text/html; charset=utf-8", 200);
}

int main(int argc, const char *argv[]) {
  struct sg_httpsrv *srv;
```

```
  uint16_t port;
  if (argc != 2) {
    printf("%s <PORT>\n", argv[0]);
    return EXIT_FAILURE;
  }
  port = strtol(argv[1], NULL, 10);
  srv = sg_httpsrv_new2(auth_cb, req_cb, err_cb, NULL);
  if (!sg_httpsrv_listen(srv, port, false)) {
    sg_httpsrv_free(srv);
    return EXIT_FAILURE;
  }
  fprintf(stdout, "Server running at http://localhost:%d\n",
          sg_httpsrv_port(srv));
  fflush(stdout);
  getchar();
  sg_httpsrv_free(srv);
  return EXIT_SUCCESS;
}
```

8.3 example_httpcomp.c

Example drastically simplified to show the basic concept of HTTP compression by deflate.

```
/*                          _
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdlib.h>
#include <stdint.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

#define PAGE                                                                   \
  "<html><head><title>Hello world</title></head><body>Hello "                  \
  "world</body></html>"
#define CONTENT_TYPE "text/html; charset=utf-8"

static void req_cb(__SG_UNUSED void *cls, struct sg_httpreq *req,
                   struct sg_httpres *res) {
  struct sg_strmap **headers;
  const char *header;
  headers = sg_httpreq_headers(req);
  if (headers) {
    header = sg_strmap_get(*headers, "Accept-Encoding");
    if (header && strstr(header, "deflate")) {
      sg_httpres_zsendbinary(res, PAGE, strlen(PAGE), CONTENT_TYPE, 200);
      return;
    }
  }
  sg_httpres_sendbinary(res, PAGE, strlen(PAGE), CONTENT_TYPE, 200);
}

int main(int argc, const char *argv[]) {
  struct sg_httpsrv *srv;
  uint16_t port;
  if (argc != 2) {
    printf("%s <PORT>\n", argv[0]);
    return EXIT_FAILURE;
  }
  port = strtol(argv[1], NULL, 10);
```

```
  srv = sg_httpsrv_new(req_cb, NULL);
  if (!sg_httpsrv_listen(srv, port, false)) {
    sg_httpsrv_free(srv);
    return EXIT_FAILURE;
  }
  fprintf(stdout, "Server running at http://localhost:%d\n",
          sg_httpsrv_port(srv));
  fflush(stdout);
  getchar();
  sg_httpsrv_free(srv);
  return EXIT_SUCCESS;
}
```

8.4 example_httpcookie.c

Simple example using server and client cookies.

```
/*
 *    ___  __ _  __ _ _   _(_)
 *   / __|/ _` |/ _` | | | | |
 *   \__ \ (_| | (_| | |_| | |
 *   |___/\__,_|\__, |\__,_|_|
 *              |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

#define CONTENT_TYPE "text/html; charset=utf-8"
#define BEGIN_PAGE "<html><head><title>Cookies</title></head><body>"
#define END_PAGE "</body></html>"
#define INITIAL_PAGE BEGIN_PAGE "Use F5 to refresh this page ..." END_PAGE
#define COUNT_PAGE BEGIN_PAGE "Refresh number: %d" END_PAGE
#define COOKIE_NAME "refresh_count"

static int strtoint(const char *str) {
  if (!str)
    return 0;
  return (int) strtol(str, NULL, 10);
}

static void req_cb(__SG_UNUSED void *cls, struct sg_httpreq *req,
                   struct sg_httpres *res) {
  struct sg_strmap **cookies;
  char str[100];
  int count;
  if (strcmp(sg_httpreq_path(req), "/favicon.ico") == 0) {
    sg_httpres_send(res, "", "", 204);
    return;
  }
  cookies = sg_httpreq_cookies(req);
  count = cookies ? strtoint(sg_strmap_get(*cookies, COOKIE_NAME)) : 0;
  if (count == 0) {
    snprintf(str, sizeof(str), INITIAL_PAGE);
    count = 1;
  } else {
    snprintf(str, sizeof(str), COUNT_PAGE, count);
    count++;
  }
  sg_httpres_send(res, str, CONTENT_TYPE, 200);
```

```
  snprintf(str, sizeof(str), "%d", count);
  sg_httpres_set_cookie(res, COOKIE_NAME, str);
}

int main(int argc, const char *argv[]) {
  struct sg_httpsrv *srv;
  uint16_t port;
  if (argc != 2) {
    printf("%s <PORT>\n", argv[0]);
    return EXIT_FAILURE;
  }
  port = strtol(argv[1], NULL, 10);
  srv = sg_httpsrv_new(req_cb, NULL);
  if (!sg_httpsrv_listen(srv, port, false)) {
    sg_httpsrv_free(srv);
    return EXIT_FAILURE;
  }
  fprintf(stdout, "Server running at http://localhost:%d\n",
          sg_httpsrv_port(srv));
  fflush(stdout);
  getchar();
  sg_httpsrv_free(srv);
  return EXIT_SUCCESS;
}
```

8.5 example_httpreq_payload.c

Simple payload example showing how to retrieve raw content (JSON, XML etc.) from the request object.

```
/*                          _
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

/*
 * Echoing payload using cURL:
 *
 * curl --header "Content-Type: application/json" --request POST --data '{"abc":123}' -w "\n" http://
     localhost:<PORT>
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

static void req_cb(__SG_UNUSED void *cls, struct sg_httpreq *req,
                   struct sg_httpres *res) {
  struct sg_str *payload = sg_httpreq_payload(req);
  sg_httpres_send(res, sg_str_content(payload), "text/plain", 200);
}

int main(int argc, const char *argv[]) {
  struct sg_httpsrv *srv;
  uint16_t port;
  if (argc != 2) {
    printf("%s <PORT>\n", argv[0]);
    return EXIT_FAILURE;
  }
```

```
  port = strtol(argv[1], NULL, 10);
  srv = sg_httpsrv_new(req_cb, NULL);
  if (!sg_httpsrv_listen(srv, port, false)) {
    sg_httpsrv_free(srv);
    return EXIT_FAILURE;
  }
  fprintf(stdout, "Server running at http://localhost:%d\n",
          sg_httpsrv_port(srv));
  fflush(stdout);
  getchar();
  sg_httpsrv_free(srv);
  return EXIT_SUCCESS;
}
```

8.6 example_httpsrv.c

Simple "hello world" HTTP server example.

```
/*                          _
 *   ___  __ _  __ _ _   _ (_)
 *  / __|/ _` |/ _` | | | || |
 *  \__ \ (_| | (_| | |_| || |
 *  |___/\__,_|\__, |\__,_|/ |
 *             |___/       |__/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

static void req_cb(__SG_UNUSED void *cls, __SG_UNUSED struct sg_httpreq *req,
                   struct sg_httpres *res) {
  sg_httpres_send(res,
                  "<html><head><title>Hello world</title></head><body>Hello "
                  "world</body></html>",
                  "text/html; charset=utf-8", 200);
}

int main(int argc, const char *argv[]) {
  struct sg_httpsrv *srv;
  uint16_t port;
  if (argc != 2) {
    printf("%s <PORT>\n", argv[0]);
    return EXIT_FAILURE;
  }
  port = strtol(argv[1], NULL, 10);
  srv = sg_httpsrv_new(req_cb, NULL);
  if (!sg_httpsrv_listen(srv, port, false)) {
    sg_httpsrv_free(srv);
    return EXIT_FAILURE;
  }
  fprintf(stdout, "Server running at http://localhost:%d\n",
          sg_httpsrv_port(srv));
  fflush(stdout);
  getchar();
  sg_httpsrv_free(srv);
  return EXIT_SUCCESS;
}
```

8.7 example_httpsrv_benchmark.c

Simple "hello world" HTTP server to be used by benchmark tests.

```
/*
 *                  _
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdlib.h>
#include <stdint.h>
#ifdef _WIN32
#include <windows.h>
#else
#include <unistd.h>
#endif
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

static unsigned int get_cpu_count(void) {
#ifdef _WIN32
#ifndef _SC_NPROCESSORS_ONLN
  SYSTEM_INFO info;
  GetSystemInfo(&info);
#define sysconf(void) info.dwNumberOfProcessors
#define _SC_NPROCESSORS_ONLN
#endif
#endif
#ifdef _SC_NPROCESSORS_ONLN
  return (unsigned int) sysconf(_SC_NPROCESSORS_ONLN);
#else
  return 0;
#endif
}

static void req_cb(__SG_UNUSED void *cls, __SG_UNUSED struct sg_httpreq *req,
                   struct sg_httpres *res) {
  sg_httpres_send(res,
                  "<html><head><title>Hello world</title></head><body>Hello "
                  "world</body></html>",
                  "text/html", 200);
}

int main(int argc, const char *argv[]) {
  struct sg_httpsrv *srv;
  unsigned int cpu_count;
  unsigned int con_limit;
  uint16_t port;
  if (argc != 2) {
    printf("%s <PORT>\n", argv[0]);
    return EXIT_FAILURE;
  }
  port = strtol(argv[1], NULL, 10);
  cpu_count = get_cpu_count();
  con_limit = 1000; /* Change to 10000 for C10K problem. */
  srv = sg_httpsrv_new(req_cb, NULL);
  sg_httpsrv_set_thr_pool_size(srv, cpu_count);
  sg_httpsrv_set_con_limit(srv, con_limit);
  if (!sg_httpsrv_listen(srv, port, false)) {
    sg_httpsrv_free(srv);
    return EXIT_FAILURE;
  }
  fprintf(stdout, "Number of processors: %d\n", cpu_count);
  fprintf(stdout, "Connections limit: %d\n", con_limit);
```

```
  fprintf(stdout, "Server running at http://localhost:%d\n",
          sg_httpsrv_port(srv));
  fflush(stdout);
  getchar();
  sg_httpsrv_free(srv);
  return EXIT_SUCCESS;
}
```

8.8 example_httpsrv_tls.c

Simple "hello world" HTTPS server example.

```
/*
 *                          _
 *   ___  __ _  __ _ _   _ (_)
 *  / __|/ _` |/ _` | | | || |
 *  \__ \ (_| | (_| | |_| || |
 *  |___/\__,_|\__, |\__,_|/ |
 *             |___/       |__/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

/*
 * Very simple insecure HTTPS server. For total security, use: https://help.ubuntu.com/community/OpenSSL.
 *
 * Client example using cURL:
 *
 * curl -k https://localhost:<PORT>
 *
 * Certificate generation:
 *
 * # Private key
 * certtool --generate-privkey --outfile server.key
 * echo 'organization = GnuTLS test server' > server.tmpl
 * echo 'cn = test.gnutls.org' >> server.tmpl
 * echo 'tls_www_server' >> server.tmpl
 * echo 'expiration_days = 3650' >> server.tmpl
 *
 * # Certificate
 * certtool --generate-self-signed --load-privkey server.key --template server.tmpl --outfile server.pem
 */

const char private_key[] =
  "-----BEGIN RSA PRIVATE KEY-----\n"
  "MIIG5QIBAAKCAYEA1qzaG2QbwnBzbRYENJbh17BoNM1XIV4PH030FJL77BR+KfL/\n"
  "qfUwY6Nkzy/iDtn1BYrUmd8Mus34obsC0ad/sIarailK//mAYVyrVHuLgOJ4c1Fu\n"
  "sqMvafEB67LLFxeItwK1X78aT9TSEbOZqpuVXmEPO0Vve53nQHaRr7zBbb0j8tBq\n"
  "xCc07yW9TRloOctholYSp1mPiaAJ5d/YIcmSEeD6NBpMZ1VguHIJb65i45EzX+Nl\n"
  "JHSVjcVgq1Ap5gg6+W2SY6TSSjbbnHANmoVMpu690jP6NXDIOY5IInnMSk+qxb4n\n"
  "wA8a/geLKooKItw85tOl27b+ap3xjwDZSrixYl5ieRakyunBipnE9bSz0dml3KiP\n"
  "M+6hbArskPufD3gjHjBhQqlqvA0imnGrn4/rwmaTFplLv3kCfyTElQcMR7ikF5ud\n"
  "Nn4yZzDzA7geWPBOZ2XeNkK5f+ULOfpvxZsYRaCSDLPiI2/wwjLvSlj6z0mmOnif\n"
  "r4NFdp/eFfPQPCoHAgMBAAECggGBAIm/N+RDhBxrk2T3r5MfDaMcqoDXEYVzmTh5\n"
  "CJj7B3MgYyP/rFUd4wLMIS9ghikJadM4ldp16PEkoNkF6nUkiSZ4Ax2HiXxeWCYh\n"
  "FD6NV6JHrwovwlwVoaLU5mqauv4CN9NWhZL+SJ/Y60I4f+2dD2cT2HYrw7EKTQxs\n"
  "CGc/Ms57gsmXOirLDYg2KxWBrAMJoYhMuoNvUE76xd8elsx3TqbyORmdllDn07wG\n"
  "UE+9Ee77iH+KpaeStzPU5oaGVZwowqHX4yaKiqOpdhy2JAKAguXu6pYhFHVdHHv/\n"
  "HRQ3bi/3P67j3rdBWdQr8Qb7EjO3ZmH24ObAfj9AN+gPhBjIxG3BGmGUT5rD4r79\n"
  "mMiR9O96pEVm+M8L+3TfN1J8S6fCgthg+vOxAqw2FEarJUHpYXT5xP7njgF2Llvk\n"
  "bJnQvQ8L1G8o4r7/wzaoAmdBmV0Yzwr5Kv4wyaLizJS3ai1gS6QsD71DLaTN1ZEn\n"
```

```
  "05tMNhCTr76Ik7Qy43mS05+upprU4QKBwQDrdynzn9EKtlTDOI1e+8wdDTlKpicV\n"
  "C/PPgZgtzluWdaGFs7emSqZzZ1xDsBbnAP1rCT0WOyvvYoGt9xNqStbTa0ahkbV6\n"
  "LBeks9zUWdXgyJ2r+a708/juW+r81ya97mtOIvCsMuvMvJnk/E1HelHtFCs7Wnxv\n"
  "syKS3IhjhrXyZGcdEPNLJdEyDrsPOZF15AMEUAqP6ZtewCIB9zn5Q1hjakkd+Mq0\n"
  "Ouh6R2GQzHXZNDJIsKiyt0EuPkrjsRydY9MCgcEA6WWJ+mtR3DLo8dV8ClIrJlkg\n"
  "LGk1rwgEgBboemB5TVN5yVuQo37Bsy7x1iVn9GCIEKOyybZUwn3xt0F2WFoAHcg5\n"
  "5hbJ94XLMUMAUmF7ML8X0rfMm2K34B4eIb6lcLGc9eULvznyAmBun+MeONOERLtF\n"
  "yRnERkZmOrk4SJohsC0uw5LoNZmT5CXKrFGL2QbQkBvOGgpFVhSGssoQx7m05gl0\n"
  "ollUaK8vltthSjfX5TKyPqFa6yB3oTLpW8oHjGR9AoHBAIfXsqMzk1UbxaDu0lh2\n"
  "6dXk2CWh37A7ugf/2vyqLZqK+Il7Gjtcm9S+T7NZNo1Eu+7xYIWf04QCj4/+l/vd\n"
  "ezxzikcSGeGG6IkDnyX/Qe2xr40UugPlcLqK2vHNajNvBgcJD1I4+mKeeCZsDGVt\n"
  "QzCET0CpvlpuvUZ+5kyM3hEeLYLOUZ4MDjlT2EU7UBj0V204hC9sdU9fhv8dUxvj\n"
  "521LVylsZ/08cvyAi+AOpPqPK2dWS6z3HiqAk5HyjvCaMwKBwQDPpRimhEhmAZ0h\n"
  "Wm9qt0PQcEahfFDYked/FeJqzd3dn8CgFiiObL1j7wYVIV5lGmSzeRAdSWwLRQWB\n"
  "pmnlnNyxomtweyHgZ1YpU5S7tiJlcf196SvNqnwwllr0ZqrFoh8k3UwgKytWVfjV\n"
  "orhGklgA1iP2EEiAxS06XYLnhMkn9mq+cLrKxQHAXqb7u+kRgnCXZUVuAWlCdiyI\n"
  "cCGRr3RznEH6FkN0hzdtnWvnHduj/AB8nA1JVq9X6PWYRhuFGfkCgcAOFpQgd2Rj\n"
  "tnRIGiGkAKR4Pyn69xdI7xrqDqdpA07rjaINBsXnGZ+xP8whfVh8JZU9VQoVzm2f\n"
  "uUwHKMogYAOnaMjeFoZ17QGS5/LVBPz9VTVD8VVY5EQr7Mh3kLUSC2h1RRDfdyE9\n"
  "RVpU2POvbfwetMVh2Q18/4i4vd02khzbn9u0JeUktVGUbVAPl6IcOPlVy9h2BseG\n"
  "8WwEjhs93VRNy/PSxmAeVYymaDSqR5eBL+/eExk+ryr93InlaQmj5Is=\n"
  "-----END RSA PRIVATE KEY-----";

const char certificate[] =
  "-----BEGIN CERTIFICATE-----\n"
  "MIIEODCCAqCgAwIBAgIMW23FvhiQ/Xip31BxMA0GCSqGSIb3DQEBCwUAMDcxGDAW\n"
  "BgNVBAMTD3Rlc3QuZ251dGxzLm9yZzEbMBkGA1UEChMSR251VExTIHRlc3Qgc2Vy\n"
  "dmVyMB4XDTE4MDgxMDE3MDUwMloXDTI4MDgwNzE3MDUwMlowNzEYMBYGA1UEAxMP\n"
  "dGVzdC5nbnV0bHMub3JnMRswGQYDVQQKExJHbnVUTFMgdGVzdCBzZXJ2ZXIwggGi\n"
  "MA0GCSqGSIb3DQEBAQUAA4IBjwAwggGKAoIBgQDWrNobZBvCcHNtFgQ0luHXsGg0\n"
  "zVchXg8fTfQUkvvsFH4p8v+p9TBjo2TPL+IO2fUFitSZ3wy6zfihuwLRp3+whqtq\n"
  "KUr/+YBhXKtUe4uA4nhzUW6yoy9p8QHrsssXF4i3ArVfvxpP1NIRs5mqm5VeYQ87\n"
  "RW97nedAdpGvvMFtvSPy0GrEJzTvJb1NGWg5y2GiVhKnWY+JoAnl39ghyZIR4Po0\n"
  "GkxnVWC4cglvrmLjkTNf42UkdJWNxWCrUCnmCDr5bZJjpNJKNtuccA2ahUym7r3S\n"
  "M/o1cMg5jkgiecxKT6rFvifADxr+B4sqigoi3Dzm06Xbtv5qnfGPANlKuLFiXmJ5\n"
  "FqTK6cGKmcT1tLPR2aXcqI8z7qFsCuyQ+58PeCMeMGFCqWq8DSKacaufj+vCZpMW\n"
  "mUu/eQJ/JMSVBwxHuKQXm502fjJnMPMDuB5Y8E5nZd42Qrl/5Qs5+m/FmxhFoJIM\n"
  "s+Ijb/DCMu9KWPrPSaY6eJ+vg0V2n94V89A8KgcCAwEAAaNEMEIwDAYDVR0TAQH/\n"
  "BAIwADATBgNVHSUEDDAKBggrBgEFBQcDATAdBgNVHQ4EFgQUxH1HUKpvYFEHrPeJ\n"
  "sY0I7HQDbIIwDQYJKoZIhvcNAQELBQADggGBABDTlhiKuuh51Rx+mpt5vjJ7zXRJ\n"
  "1RoHY92AmZY49hfdpUp6mMLvbRdD6REv4pelIORGsqgPk4MPCQsaFGbpZS1CokLz\n"
  "Sex4LZruHYWtv18fUFliJIZZSITnArNB29lXAem5T20D04bCCYLJJB3VTcPilbkf\n"
  "ipT/h1CyhbWX14ZtkzzpWMAwLgod6uZvJqJXTpjwdWA7Anp4yfh2QxBYC5/us4xP\n"
  "wHa0euWOBZ+Q9ZNZ/fFdLESLSbBob9736hBg1NSgBFCMNezqs18/EGIJcS7w96PN\n"
  "YJtVsVhcQJMMT4dnaSM/Ri4CPv7j8/zll6uq4kHpxZLhuxeRSeuKBn6jl0wHQFd1\n"
  "7bpHrRLBuRyDhPWrzdmMY2dyJ5DkO39auisAyJza8IddfNnCa7howSjp/ZvZN9Sf\n"
  "gi1klZeSpe+iijWQaxjIKAr/g8Rn+ALfeMAitm6DjCcTUkXdKtXVqTwdFZRNxrNH\n"
  "lqt+H07raUsv/p50oVS6/Euv8fBm3EKPwxC64w==\n"
  "-----END CERTIFICATE-----";

static void req_cb(__SG_UNUSED void *cls, __SG_UNUSED struct sg_httpreq *req,
                   struct sg_httpres *res) {
  sg_httpres_send(res,
                  "<html><head><title>Hello world</title></head><body>Hello "
                  "<font color=\"green\">HTTPS</font></body></html>",
                  "text/html; charset=utf-8", 200);
}

int main(int argc, const char *argv[]) {
  struct sg_httpsrv *srv;
  uint16_t port;
  if (argc != 2) {
    printf("%s <PORT>\n", argv[0]);
    return EXIT_FAILURE;
  }
  port = strtol(argv[1], NULL, 10);
  srv = sg_httpsrv_new(req_cb, NULL);
  if (!sg_httpsrv_tls_listen(srv, private_key, certificate, port, false)) {
    sg_httpsrv_free(srv);
    return EXIT_FAILURE;
  }
  fprintf(stdout, "Server running at https://localhost:%d\n",
          sg_httpsrv_port(srv));
  fflush(stdout);
  getchar();
  sg_httpsrv_free(srv);
  return EXIT_SUCCESS;
}
```

8.9 example_httpsrv_tls_cert_auth.c

Simple client-side certificate authentication using GnuTLS.

```
/*
 *                      _
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <errno.h>
#include <gnutls/gnutls.h>
#include <gnutls/x509.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

/*
 * Simple example using TLS client authentication.
 *
 * Client example using cURL:
 *
 * curl -k --cert certs/client.p12 --pass abc123 --cert-type p12 https://localhost:<PORT>
 *
 * Certificate generation:
 *
 * ## CA
 * certtool --generate-privkey --outfile ca.key
 * echo 'cn = GnuTLS test CA' > ca.tmpl
 * echo 'ca' >> ca.tmpl
 * echo 'cert_signing_key' >> ca.tmpl
 * echo 'expiration_days = 3650' >> ca.tmpl
 * certtool --generate-self-signed --load-privkey ca.key --template ca.tmpl --outfile ca.pem
 *
 * ## Server
 * certtool --generate-privkey --outfile server.key
 * echo 'organization = GnuTLS test server' > server.tmpl
 * echo 'cn = test.gnutls.org' >> server.tmpl
 * echo 'tls_www_server' >> server.tmpl
 * echo 'expiration_days = 3650' >> server.tmpl
 * certtool --generate-certificate --load-ca-privkey ca.key --load-ca-certificate ca.pem --load-privkey
       server.key --template server.tmpl --outfile server.pem
 *
 * ## Client
 * certtool --generate-privkey --outfile client.key
 * echo 'cn = GnuTLS test client' > client.tmpl
 * echo 'tls_www_client' >> client.tmpl
 * echo 'expiration_days = 3650' >> client.tmpl
 * certtool --generate-certificate --load-ca-certificate ca.pem --load-ca-privkey ca.key --load-privkey
       client.key --template client.tmpl --outfile client.pem
 * certtool --to-p12 --p12-name=MyKey --password=abc123 --load-ca-certificate ca.pem --load-privkey
       client.key --load-certificate client.pem --outder --outfile client.p12
 */

#define KEY_FILE SG_EXAMPLES_CERTS_DIR "/server.key"
#define CERT_FILE SG_EXAMPLES_CERTS_DIR "/server.pem"
#define CA_FILE SG_EXAMPLES_CERTS_DIR "/ca.pem"

#define ERR_SIZE 256
#define PAGE_FMT                                                               \
  "<html><head><title>Hello world</title></head><body><font "                  \
  "color=\"%s\">%s</font></font></body></html>"
#define SECRET_MSG "Secret"

static void concat(char *s1, ...) {
  va_list ap;
  const char *s;
  va_start(ap, s1);
```

```
  while ((s = va_arg(ap, const char *)))
    strcat(s1, s);
  va_end(ap);
}

static bool sess_verify_cert(gnutls_session_t tls_session,
                             const char *line_break, char *err) {
  gnutls_x509_crt_t cert = NULL;
  const gnutls_datum_t *certs;
  size_t len;
  unsigned int status, certs_size;
  int ret;
  if (!tls_session || !line_break || !err) {
    sg_strerror(EINVAL, err, ERR_SIZE);
    return false;
  }
  if ((ret = gnutls_certificate_verify_peers2(tls_session, &status)) !=
      GNUTLS_E_SUCCESS) {
    concat(err, "Error verifying peers: ", gnutls_strerror(ret), line_break,
           NULL);
    goto error;
  }
  if (status & GNUTLS_CERT_INVALID)
    concat(err, "The certificate is not trusted", line_break, NULL);
  if (status & GNUTLS_CERT_SIGNER_NOT_FOUND)
    concat(err, "The certificate has not got a known issuer", line_break, NULL);
  if (status & GNUTLS_CERT_REVOKED)
    concat(err, "The certificate has been revoked", line_break, NULL);
  if (gnutls_certificate_type_get(tls_session) != GNUTLS_CRT_X509) {
    concat(err, "The certificate type is not X.509", line_break, NULL);
    goto error;
  }
  if ((ret = gnutls_x509_crt_init(&cert)) != GNUTLS_E_SUCCESS) {
    concat(err,
           "Error in the certificate initialization: ", gnutls_strerror(ret),
           line_break, NULL);
    goto error;
  }
  if (!(certs = gnutls_certificate_get_peers(tls_session, &certs_size))) {
    concat(err, "No certificate was found", line_break, NULL);
    goto error;
  }
  if ((ret = gnutls_x509_crt_import(cert, &certs[0], GNUTLS_X509_FMT_DER)) !=
      GNUTLS_E_SUCCESS) {
    concat(err, "Error parsing certificate: ", gnutls_strerror(ret), line_break,
           NULL);
    goto error;
  }
  if (gnutls_x509_crt_get_expiration_time(cert) < time(NULL)) {
    concat(err, "The certificate has expired", line_break, NULL);
    goto error;
  }
  if (gnutls_x509_crt_get_activation_time(cert) > time(NULL)) {
    concat(err, "The certificate has not been activated yet", line_break, NULL);
    goto error;
  }
error:
  len = strlen(err);
  err[len - strlen("<br>")] = '\0';
  gnutls_x509_crt_deinit(cert);
  return len == 0;
}

static void req_cb(__SG_UNUSED void *cls, struct sg_httpreq *req,
                   struct sg_httpres *res) {
  char msg[ERR_SIZE];
  char *color, *page;
  size_t page_size;
  unsigned int status;
  if (sess_verify_cert(sg_httpreq_tls_session(req), "<br>", msg)) {
    strcpy(msg, SECRET_MSG);
    color = "green";
    status = 200;
  } else {
    color = "red";
    status = 500;
  }
  page_size = (size_t) snprintf(NULL, 0, PAGE_FMT, color, msg);
  page = sg_alloc(page_size);
  snprintf(page, page_size, PAGE_FMT, color, msg);
  sg_httpres_send(res, page, "text/html; charset=utf-8", status);
  sg_free(page);
}

int main(int argc, const char *argv[]) {
  struct sg_httpsrv *srv;
  gnutls_datum_t key_file, cert_file, ca_file;
```

```
  int ret, status;
  uint16_t port;
  if (argc != 2) {
    printf("%s <PORT>\n", argv[0]);
    return EXIT_FAILURE;
  }
  port = strtol(argv[1], NULL, 10);
  status = EXIT_FAILURE;
  srv = sg_httpsrv_new(req_cb, NULL);
  memset(&key_file, 0, sizeof(gnutls_datum_t));
  memset(&cert_file, 0, sizeof(gnutls_datum_t));
  memset(&ca_file, 0, sizeof(gnutls_datum_t));
  if ((ret = gnutls_load_file(KEY_FILE, &key_file)) != GNUTLS_E_SUCCESS) {
    fprintf(stderr, "Error loading the private key \"%s\": %s\n", KEY_FILE,
            gnutls_strerror(ret));
    fflush(stdout);
    goto error;
  }
  if ((ret = gnutls_load_file(CERT_FILE, &cert_file)) != GNUTLS_E_SUCCESS) {
    fprintf(stderr, "Error loading the certificate \"%s\": %s\n", CERT_FILE,
            gnutls_strerror(ret));
    fflush(stdout);
    goto error;
  }
  if ((ret = gnutls_load_file(CA_FILE, &ca_file)) != GNUTLS_E_SUCCESS) {
    fprintf(stderr, "Error loading the CA \"%s\": %s\n", CA_FILE,
            gnutls_strerror(ret));
    fflush(stdout);
    goto error;
  }
  if (sg_httpsrv_tls_listen2(srv, (const char *) key_file.data, NULL,
                             (const char *) cert_file.data,
                             (const char *) ca_file.data, NULL, port, false)) {
    status = EXIT_SUCCESS;
    fprintf(stdout, "Server running at https://localhost:%d\n",
            sg_httpsrv_port(srv));
    fflush(stdout);
    getchar();
  }
error:
  sg_httpsrv_free(srv);
  if (key_file.size > 0)
    gnutls_free(key_file.data);
  if (cert_file.size > 0)
    gnutls_free(cert_file.data);
  if (ca_file.size > 0)
    gnutls_free(ca_file.data);
  return status;
}
```

8.10 example_httpuplds.c

Simple example showing how to upload files to the server.

```
/*
 *                          _
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
```

```c
#include <limits.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

#ifdef _WIN32
#define PATH_SEP '\\'
#else
#define PATH_SEP '/'
#endif

#define PAGE_FORM                                                              \
  "<html>"                                                                     \
  "<body>"                                                                     \
  "<form action=\"\" method=\"post\" enctype=\"multipart/form-data\">"         \
  "<fieldset>"                                                                 \
  "<legend>Choose the files:</legend>"                                         \
  "File 1: <input type=\"file\" name=\"file1\"/><br>"                          \
  "File 2: <input type=\"file\" name=\"file2\"/><br>"                          \
  "<input type=\"submit\"/>"                                                   \
  "</fieldset>"                                                                \
  "</form>"                                                                    \
  "</body>"                                                                    \
  "</html>"

#define PAGE_DONE                                                              \
  "<html>"                                                                     \
  "<head>"                                                                     \
  "<title>Uploads</title>"                                                     \
  "</head>"                                                                    \
  "<body>"                                                                     \
  "<strong>Uploaded files:</strong><br>"                                       \
  "%s"                                                                         \
  "</body>"                                                                    \
  "</html>"

#define CONTENT_TYPE "text/html; charset=utf-8"

static void process_uploads(struct sg_httpreq *req, struct sg_httpres *res) {
  struct sg_httpupld *upld;
  struct sg_str *body;
  const char *name;
  char *str;
  char errmsg[256];
  int errnum;
  body = sg_str_new();
  sg_str_printf(body, "<ol>");
  upld = sg_httpreq_uploads(req);
  while (upld) {
    name = sg_httpupld_name(upld);
    errnum = sg_httpupld_save(upld, true);
    if (errnum == 0)
      sg_str_printf(body, "<li><a href=\"?file=%s\">%s</a></li>", name, name);
    else {
      sg_strerror(errnum, errmsg, sizeof(errmsg));
      sg_str_printf(body, "<li><font color='red'>%s - failed - %s</font></li>",
                    name, errmsg);
    }
    sg_httpuplds_next(&upld);
  }
  sg_str_printf(body, "</ol>");
  str = strdup(sg_str_content(body));
  sg_str_clear(body);
  sg_str_printf(body, PAGE_DONE, str);
  free(str);
  sg_httpres_send(res, sg_str_content(body), CONTENT_TYPE, 200);
  sg_str_free(body);
}

static void req_cb(__SG_UNUSED void *cls, struct sg_httpreq *req,
                   struct sg_httpres *res) {
  struct sg_strmap **qs;
  const char *file;
  char path[PATH_MAX];
  if (sg_httpreq_is_uploading(req))
    process_uploads(req, res);
  else {
    qs = sg_httpreq_params(req);
    if (qs) {
      file = sg_strmap_get(*qs, "file");
      if (file) {
        sprintf(path, "%s%c%s", sg_tmpdir(), PATH_SEP, file);
        sg_httpres_download(res, path);
      }
    } else
      sg_httpres_send(res, PAGE_FORM, CONTENT_TYPE, 200);
  }
```

```
}

int main(int argc, const char *argv[]) {
  struct sg_httpsrv *srv;
  uint16_t port;
  if (argc != 2) {
    printf("%s <PORT>\n", argv[0]);
    return EXIT_FAILURE;
  }
  port = strtol(argv[1], NULL, 10);
  srv = sg_httpsrv_new(req_cb, NULL);
  if (!sg_httpsrv_listen(srv, port, false)) {
    sg_httpsrv_free(srv);
    return EXIT_FAILURE;
  }
  fprintf(stdout, "Server running at http://localhost:%d\n",
          sg_httpsrv_port(srv));
  fflush(stdout);
  getchar();
  sg_httpsrv_free(srv);
  return EXIT_SUCCESS;
}
```

8.11 example_router_segments.c

Simple example showing how to access the path segments of the router feature.

```
/*
 *                                _
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdlib.h>
#include <stdio.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

static int segments_iter_cb(__SG_UNUSED void *cls, unsigned int index,
                            const char *segment) {
  fprintf(stdout, " %d: %s\n", index, segment);
  return 0;
}

static void route_cb(void *cls, struct sg_route *route) {
  fprintf(stdout, "%s: %s\n", sg_route_path(route), (const char *) cls);
  sg_route_segments_iter(route, segments_iter_cb, NULL);
}

int main(void) {
  struct sg_router *router;
  struct sg_route *routes = NULL;
  sg_routes_add(&routes, "/foo/[0-9]+", route_cb, "foo-data");
  sg_routes_add(&routes, "/bar/([a-zA-Z]+)/([0-9]+)", route_cb, "bar-data");
  router = sg_router_new(routes);
  sg_router_dispatch(router, "/foo/123", NULL);
  fprintf(stdout, "---\n");
  sg_router_dispatch(router, "/bar/abc/123", NULL);
  sg_routes_cleanup(&routes);
  sg_router_free(router);
  fflush(stdout);
  return EXIT_SUCCESS;
```

```
}
```

8.12 example_router_simple.c

Simple example showing how to use the router feature.

```
/*
 *                      _
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdlib.h>
#include <stdio.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

static void route_cb(void *cls, struct sg_route *route) {
  fprintf(stdout, "%s: %s\n", sg_route_path(route), (const char *) cls);
  fflush(stdout);
}

int main(void) {
  struct sg_router *router;
  struct sg_route *routes = NULL;
  sg_routes_add(&routes, "/foo", route_cb, "foo-data");
  sg_routes_add(&routes, "/bar", route_cb, "bar-data");
  router = sg_router_new(routes);
  sg_router_dispatch(router, "/foo", NULL);
  sg_routes_cleanup(&routes);
  sg_router_free(router);
  return EXIT_SUCCESS;
}
```

8.13 example_router_srv.c

Simple example showing how to use the router feature in a HTTP server.

```
/*
 *                      _
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
```

```
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

/*
 * Tests:
 *
 * # return "Home"
 * curl http://localhost:<PORT>/home
 * # return "Download"
 * curl http://localhost:<PORT>/download
 * # return "file: <FILENAME>"
 * curl http://localhost:<PORT>/download/<FILENAME>
 * # return "About"
 * curl http://localhost:<PORT>/about
 * # return "404"
 * curl http://localhost:<PORT>/other
 */

#include <stdio.h>
#include <stdlib.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

struct holder {
  struct sg_httpreq *req;
  struct sg_httpres *res;
};

static int route_download_file_cb(void *cls, const char *name,
                                  const char *val) {
  sprintf(cls, "%s: %s", name, val);
  return 0;
}

static void route_home_cb(__SG_UNUSED void *cls, struct sg_route *route) {
  struct holder *holder = sg_route_user_data(route);
  sg_httpres_send(
    holder->res,
    "<html><head><title>Home</title></head><body>Home</body></html>",
    "text/html", 200);
}

static void route_download_cb(__SG_UNUSED void *cls, struct sg_route *route) {
  struct holder *holder = sg_route_user_data(route);
  struct sg_str *page = sg_str_new();
  char file[256];
  memset(file, 0, sizeof(file));
  sg_route_vars_iter(route, route_download_file_cb, file);
  if (strlen(file) == 0)
    strcpy(file, "Download");
  sg_str_printf(
    page, "<html><head><title>Download</title></head><body>%s</body></html>",
    file);
  sg_httpres_send(holder->res, sg_str_content(page), "text/html", 200);
  sg_str_free(page);
}

static void route_about_cb(__SG_UNUSED void *cls, struct sg_route *route) {
  struct holder *holder = sg_route_user_data(route);
  sg_httpres_send(
    holder->res,
    "<html><head><title>About</title></head><body>About</body></html>",
    "text/html", 200);
}

static void req_cb(__SG_UNUSED void *cls, struct sg_httpreq *req,
                   struct sg_httpres *res) {
  struct sg_router *router = cls;
  struct holder holder = {req, res};
  if (sg_router_dispatch(router, sg_httpreq_path(req), &holder) != 0)
    sg_httpres_send(
      res, "<html><head><title>Not found</title></head><body>404</body></html>",
      "text/html", 404);
}

int main(void) {
  struct sg_route *routes = NULL;
  struct sg_router *router;
  struct sg_httpsrv *srv;
  sg_routes_add(&routes, "/home", route_home_cb, NULL);
  sg_routes_add(&routes, "/download", route_download_cb, NULL);
  sg_routes_add(&routes, "/download/(?P<file>[a-z]+)", route_download_cb, NULL);
  sg_routes_add(&routes, "/about", route_about_cb, NULL);
```

```
  router = sg_router_new(routes);
  srv = sg_httpsrv_new(req_cb, router);
  if (!sg_httpsrv_listen(srv, 0 /* 0 = port chosen randomly */, false)) {
    sg_httpsrv_free(srv);
    return EXIT_FAILURE;
  }
  fprintf(stdout, "Server running at http://localhost:%d\n",
          sg_httpsrv_port(srv));
  fflush(stdout);
  getchar();
  sg_httpsrv_free(srv);
  sg_routes_cleanup(&routes);
  sg_router_free(router);
  return EXIT_SUCCESS;
}
```

8.14 example_router_vars.c

Simple example showing how to access the path variables of the router feature.

```
/*                          _
 *   ___  __ _  __ _ _   _ (_)
 *  / __|/ _` |/ _` | | | || |
 *  \__ \ (_| | (_| | |_| || |
 *  |___/\__,_|\__, |\__,_|/ |
 *             |___/       |__/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

#include <stdlib.h>
#include <stdio.h>
#include <sagui.h>

/* NOTE: Error checking has been omitted to make it clear. */

static int vars_iter_cb(__SG_UNUSED void *cls, const char *name,
                        const char *val) {
  fprintf(stdout, " %s: %s\n", name, val);
  return 0;
}

static void route_cb(void *cls, struct sg_route *route) {
  fprintf(stdout, "%s: %s\n", sg_route_path(route), (const char *) cls);
  sg_route_vars_iter(route, vars_iter_cb, NULL);
}

int main(void) {
  struct sg_router *router;
  struct sg_route *routes = NULL;
  sg_routes_add(&routes, "/foo/bar", route_cb, "foo-bar-data");
  sg_routes_add(&routes, "/bar", route_cb, "bar-data");
  sg_routes_add(&routes, "/customer/(?P<name>[a-zA-Z]+)", route_cb,
                "customer-data");
  sg_routes_add(&routes, "/product/(?P<id>[0-9]+)", route_cb, "product-data");
  sg_routes_add(&routes, "/employee/(?P<id>[0-9]+)/[a|i]", route_cb,
                "employee-data");
  router = sg_router_new(routes);
  sg_router_dispatch(router, "/foo/bar", NULL);
  fprintf(stdout, "---\n");
  sg_router_dispatch(router, "/customer/Torvalds", NULL);
  fprintf(stdout, "---\n");
  sg_router_dispatch(router, "/product/123", NULL);
  fprintf(stdout, "---\n");
  sg_router_dispatch(router, "/employee/123/i", NULL);
  sg_routes_cleanup(&routes);
  sg_router_free(router);
```

```
  fflush(stdout);
  return EXIT_SUCCESS;
}
```

8.15 example_str.c

Simple example showing the sg_str feature.

```
/*
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

/* NOTE: Error checking has been omitted to make it clear. */

#include <stdio.h>
#include <stdlib.h>
#include <sagui.h>

int main(void) {
  struct sg_str *str = sg_str_new();
  sg_str_printf(str, "%s %s", "Hello", "world");
  printf("%s", sg_str_content(str));
  sg_str_free(str);
  return EXIT_SUCCESS;
}
```

8.16 example_strmap.c

Simple example showing the sg_strmap feature.

```
/*
 *   ___  __ _  __ _ _   _(_)
 *  / __|/ _` |/ _` | | | | |
 *  \__ \ (_| | (_| | |_| | |
 *  |___/\__,_|\__, |\__,_|_|
 *             |___/
 *
 * Cross-platform library which helps to develop web servers or frameworks.
 *
 * Copyright (C) 2016-2019 Silvio Clecio <silvioprog@gmail.com>
 *
 * Sagui library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Lesser General Public
 * License as published by the Free Software Foundation; either
 * version 2.1 of the License, or (at your option) any later version.
 *
 * Sagui library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public
 * License along with Sagui library; if not, write to the Free Software
 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 */

/* NOTE: Error checking has been omitted to make it clear. */
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sagui.h>

static int map_sort(__SG_UNUSED void *cls, struct sg_strmap *pair_a,
                    struct sg_strmap *pair_b) {
  return strcmp(sg_strmap_val(pair_b), sg_strmap_val(pair_a)); /* desc */
}

static int map_iter(__SG_UNUSED void *cls, struct sg_strmap *pair) {
  const char *name = sg_strmap_name(pair);
  printf("\t%c: %s\n", *name, name);
  return 0;
}

static void chat(struct sg_strmap **map, const char *name, const char *msg) {
  struct sg_strmap *pair;
  sg_strmap_set(map, name, msg);
  if (msg && (sg_strmap_find(*map, name, &pair) == 0))
    printf("%c:\t%s\n", *sg_strmap_name(pair), sg_strmap_val(pair));
}

int main(void) {
  struct sg_strmap *map = NULL;
  chat(&map, "Clecio", "Hello!");
  chat(&map, "Paim", "Hello. How are you?");
  chat(&map, "Clecio", "I'm fine. And you?");
  chat(&map, "Paim", "Me too.");
  printf("\nChatters:\n");
  sg_strmap_sort(&map, &map_sort, NULL);
  sg_strmap_iter(map, &map_iter, NULL);
  sg_strmap_cleanup(&map);
  return EXIT_SUCCESS;
}
```

Index

Path routing, 53

String, 11
String map, 14

Utilities, 4