lhings-c

Generated by Doxygen 1.8.6

Fri Oct 23 2015 07:20:28

Contents

1 Data Structure Index 1

1.1 Data Structures . . . 1

2 File Index 1

2.1 File List . . . 2

3 Data Structure Documentation 2

3.1 _action Struct Reference . . . 2

3.1.1 Detailed Description . . . 2

3.2 _device Struct Reference . . . 2

3.2.1 Detailed Description . . . 3

3.3 _device_info Struct Reference . . . 3

3.3.1 Detailed Description . . . 3

3.4 _event Struct Reference . . . 3

3.4.1 Detailed Description . . . 4

3.5 _lh_config Struct Reference . . . 4

3.6 _status_component Struct Reference . . . 4

3.6.1 Detailed Description . . . 4

3.7 collection_item Struct Reference . . . 4

3.8 dictionary Struct Reference . . . 5

3.9 dictionary_item Struct Reference . . . 5

3.10 http_response Struct Reference . . . 5

3.10.1 Detailed Description . . . 5

3.11 list Struct Reference . . . 5

3.12 string Struct Reference . . . 6

3.13 StunAttribute Struct Reference . . . 6

3.14 StunMessage Struct Reference . . . 6

3.14.1 Detailed Description . . . 6

4 File Documentation 6

4.1 abstraction/http-comm/http_api.h File Reference . . . 6

4.1.1 Detailed Description . . . 7

4.1.2 Function Documentation . . . 7

4.2 abstraction/permanent-storage/storage_api.h File Reference . . . 8

4.2.1 Detailed Description . . . 9

4.2.2 Function Documentation . . . 9

4.3 abstraction/timing/lhings_time.h File Reference . . . 9

4.3.1 Detailed Description . . . 10

4.3.2 Function Documentation . . . 10

4.4 abstraction/udp-comm/udp_api.h File Reference 11
4.4.1 Detailed Description 11
4.4.2 Function Documentation 11
4.5 core/lhings.h File Reference 12
4.5.1 Detailed Description 13
4.5.2 Typedef Documentation 14
4.5.3 Function Documentation 15
4.5.4 Variable Documentation 17

Index **19**

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

_action
LH_Action stores the details of an action, like its name and arguments 2

_device
LH_Device is the main data type used to store the data of a device and its capabilities 2

_device_info
LH_Info contains generic information about the device 3

_event
LH_Event stores the details of an event, like its name and its components, if any 3

_lh_config 4

_status_component
LH_Component is used to define Lhings `typed parameters` 4

collection_item 4

dictionary 5

dictionary_item 5

http_response
Data type that encapsulates an HTTP response 5

list 5

string 6

StunAttribute 6

StunMessage
STUN message data structure 6

2 File Index

2.1 File List

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

abstraction/http-comm/http_api.h
This file defines all the functions related to HTTP communication 6

abstraction/permanent-storage/storage_api.h
Contains an API that abstracts the access to permanent storage 8

abstraction/timing/lhings_time.h
This header file defines the functions that provide access to the hardware clock of the device 9

abstraction/udp-comm/udp_api.h
This header file contains all the functions that abstract the UDP communications of the library 11

core/lhings.h
Main header file for the Lhings C library 12

3 Data Structure Documentation

3.1 _action Struct Reference

LH_Action stores the details of an action, like its name and arguments.

`#include <lhings.h>`

Data Fields

- char * **name**
- char * **description**
- LH_List * **arguments**
- void(* action_function)(LH_Dict *argument_values)

  A pointer to the function that actually performs the action.

3.1.1 Detailed Description

LH_Action stores the details of an action, like its name and arguments.

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

- core/lhings.h

3.2 _device Struct Reference

LH_Device is the main data type used to store the data of a device and its capabilities.

`#include <lhings.h>`

Data Fields

- char * **name**
- char * **type**
- char * **uuid**
- char * **api_key**

- char * **username**
- LH_Info **info**
- LH_List * status_components

  A list of LH_Component that contains the information of the status components of the device.
- LH_List * events

  A list of LH_Event that contains all the events that can be notified by the device.
- LH_List * actions

  A list of LH_Action that contains all the information regarding the actions that can be performed by the device.

3.2.1 Detailed Description

LH_Device is the main data type used to store the data of a device and its capabilities.

It also stores the information about the Lhings account to which the device belongs.

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

- core/lhings.h

3.3 _device_info Struct Reference

LH_Info contains generic information about the device.

```
#include <lhings.h>
```

Data Fields

- char * **model_name**
- char * **manufacturer**
- char * **device_type**
- char * **serial_number**

3.3.1 Detailed Description

LH_Info contains generic information about the device.

This information will be added to the device descriptor.

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

- core/lhings.h

3.4 _event Struct Reference

LH_Event stores the details of an event, like its name and its components, if any.

```
#include <lhings.h>
```

Data Fields

- char * **name**
- LH_List * **components**

3.4.1 Detailed Description

LH_Event stores the details of an event, like its name and its components, if any.

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

- core/lhings.h

3.5 _lh_config Struct Reference

Data Fields

- float **loop_frequency_millis**

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

- core/lhings.h

3.6 _status_component Struct Reference

LH_Component is used to define Lhings `typed parameters`.

`#include <lhings.h>`

Data Fields

- char * name
  The name of the component.
- LH_ComponentType type
  The component type.
- void * value
  A pointer to the variable that contains the value of this component.

3.6.1 Detailed Description

LH_Component is used to define Lhings `typed parameters`.

Typed parameters are used in 3 places in the Lhings library:

- To define status components of the device in the call to lh_model_add_status_component().
- To define event components of the device in the call to lh_model_add_event().
- To define arguments of the actions in the call to lh_model_add_action().

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

- core/lhings.h

3.7 collection_item Struct Reference

Data Fields

- void * **ptr_this**

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

- core/utils/data_structures.h

3.8 dictionary Struct Reference

Data Fields

- LH_DictItem * **items**
- uint16_t **size**
- uint16_t **max_capacity**

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

- core/utils/data_structures.h

3.9 dictionary_item Struct Reference

Data Fields

- char * **ptr_key**
- void * **ptr_value**

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

- core/utils/data_structures.h

3.10 http_response Struct Reference

Data type that encapsulates an HTTP response.

`#include <http_api.h>`

Data Fields

- char * response_body
  Body of the response.
- uint16_t * http_code
  HTTP status code of the response.

3.10.1 Detailed Description

Data type that encapsulates an HTTP response.

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

- abstraction/http-comm/http_api.h

3.11 list Struct Reference

Data Fields

- LH_ListItem * **items**
- uint16_t **size**
- uint16_t **max_capacity**

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

- core/utils/data_structures.h

3.12 string Struct Reference

Data Fields

- char * **ptr**
- size_t **len**

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

- abstraction/http-comm/http_api.c

3.13 StunAttribute Struct Reference

Data Fields

- const uint8_t * **bytes**
- uint16_t **length**
- uint16_t **attr_type**

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

- core/stun-messaging/stun_message.h

3.14 StunMessage Struct Reference

STUN message data structure.

```
#include <stun_message.h>
```

Data Fields

- uint8_t * **bytes**
- uint16_t **length**
- uint16_t **class**
- uint16_t **method**
- char * **api_key**

3.14.1 Detailed Description

STUN message data structure.

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

- core/stun-messaging/stun_message.h

4 File Documentation

4.1 abstraction/http-comm/http_api.h File Reference

This file defines all the functions related to HTTP communication.

```
#include <stdint.h>
#include "../../core/utils/data_structures.h"
```

Data Structures

- struct http_response

  Data type that encapsulates an HTTP response.

Macros

- #define **PERFORM_GET** 0
- #define **PERFORM_POST** 1
- #define **PERFORM_PUT** 2
- #define **HTTP_OK** 200
- #define **HTTP_CREATED** 201

Typedefs

- typedef struct http_response LH_HttpResponse

  Data type that encapsulates an HTTP response.

Functions

- LH_HttpResponse ∗ lh_http_execute_get (const char ∗url, LH_Dict ∗headers)

  Performs an HTTP GET request to the given url, sending the given headers to the server.
- LH_HttpResponse ∗ lh_http_execute_post (const char ∗url, LH_Dict ∗headers, const char ∗post_body)

  Performs an HTTP POST request to the given url, sending the given headers to the server, and the given request body.
- LH_HttpResponse ∗ lh_http_execute_put (const char ∗url, LH_Dict ∗headers, const char ∗put_body)

  Performs an HTTP PUT request to the given url, sending the given headers to the server, and the given request body.
- void lh_http_free (LH_HttpResponse ∗response)

  Function used to free the memory allocated for LH_HttpResponse∗ by any of the `lh_http_execute_∗` functions.

4.1.1 Detailed Description

This file defines all the functions related to HTTP communication. The HTTP API defines a set of generic functions used by the Lhings C library to perform all kind of HTTP request.

All the functions in this header file belong to the abstraction API of the library and need to be reimplemented when changing platform. The documentation of each function contains all the information about its expected behaviour. This information must be carefully followed when porting the library to other platforms.

4.1.2 Function Documentation

4.1.2.1 LH_HttpResponse∗ lh_http_execute_get (const char ∗ *url*, LH_Dict ∗ *headers*)

Performs an HTTP GET request to the given url, sending the given headers to the server.

Stores the HTTP code of the response and the response body in a struct LH_HttpResponse.

Parameters

url	The url of the request.
headers	An LH_Dict containing the key value pairs of the headers (for instance key "Content-type", value "application/json").

Returns

A pointer to LH_HttpResponse containing the HTTP code and body of the response.

4.1.2.2 LH_HttpResponse* lh_http_execute_post (const char * *url*, LH_Dict * *headers*, const char * *post_body*)

Performs an HTTP POST request to the given url, sending the given headers to the server, and the given request body.

Stores the HTTP code of the response and the response body in a struct LH_HttpResponse.

Parameters

url	The url of the request.
headers	An LH_Dict containing the key value pairs of the headers (for instance key "Content-type", value "application/json").
post_body	A string containing the request body.

Returns

A pointer to LH_HttpResponse containing the HTTP code and body of the response.

4.1.2.3 LH_HttpResponse* lh_http_execute_put (const char * *url*, LH_Dict * *headers*, const char * *put_body*)

Performs an HTTP PUT request to the given url, sending the given headers to the server, and the given request body.

Stores the HTTP code of the response and the response body in a struct LH_HttpResponse.

Parameters

url	The url of the request.
headers	An LH_Dict containing the key value pairs of the headers (for instance key "Content-type", value "application/json").
post_body	A string containing the request body.

Returns

A pointer to LH_HttpResponse containing the HTTP code and body of the response.

4.1.2.4 void lh_http_free (LH_HttpResponse * *response*)

Function used to free the memory allocated for LH_HttpResponse* by any of the `lh_http_execute_*` functions.

Parameters

response	

4.2 abstraction/permanent-storage/storage_api.h File Reference

Contains an API that abstracts the access to permanent storage.

Functions

- char * lh_storage_get_uuid (char *requested_device_name)

  Searchs in the permanent storage for the uuid of a device with the given name.
- int lh_storage_save_uuid (const char *device_name, const char *uuid)

  Saves the uuid of the device with the given name in the permanent storage, so that it can be retrieved later using lh_storage_get_uuid.

4.2.1 Detailed Description

Contains an API that abstracts the access to permanent storage. Basically this API must provide the functionality needed to store the uuid of the device and its name between device executions.

All the functions in this header file belong to the abstraction API of the library and need to be reimplemented when changing platform. The documentation of each function contains all the information about its expected behaviour. This information must be carefully followed when porting the library to other platforms.

4.2.2 Function Documentation

4.2.2.1 char* lh_storage_get_uuid (char * *requested_device_name*)

Searchs in the permanent storage for the uuid of a device with the given name.

Parameters

requested_device_name	The name of the device whose uuid is needed.

Returns

A pointer to a string containing the uuid of the device or a null pointer if it could not be found.

4.2.2.2 int lh_storage_save_uuid (const char * *device_name*, const char * *uuid*)

Saves the uuid of the device with the given name in the permanent storage, so that it can be retrieved later using lh_storage_get_uuid.

Any storage API and/or hardware can be used, as long as it allows to store the pairs device name - uuid, in a way that they can be searched and retrieved later (even after device reboot or shutdown).

Parameters

device_name	A pointer to a string containing the name of the device.
uuid	A pointer to a string containing the uuid of the device. The uuid will always be a string like xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx, where x can be any hexadecimal digit ([0-9abcdef]).

Returns

1 if the pair name, uuid could be successfully saved, 0 otherwise.

4.3 abstraction/timing/lhings_time.h File Reference

This header file defines the functions that provide access to the hardware clock of the device.

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

Functions

- uint32_t lh_get_UTC_unix_time ()

  Returns a 32 bit integer with the UNIX time in seconds.
- void lh_update_time_offset (uint32_t server_time)

  Updates an internal time reference so that the values returned by lh_get_UTC_unix_time are synchronized with the time of the server.
- void lh_sleep (uint32_t millis)

  Stops the execution of the program during a given period of time.

- uint32_t lh_get_absolute_time_millis ()

  Function used for precision timing.

4.3.1 Detailed Description

This header file defines the functions that provide access to the hardware clock of the device. All the functions in this header file belong to the abstraction API of the library and need to be reimplemented when changing platform. The documentation of each function contains all the information about its expected behaviour. This information must be carefully followed when porting the library to other platforms.

4.3.2 Function Documentation

4.3.2.1 uint32_t lh_get_absolute_time_millis ( )

Function used for precision timing.

Returns the time elapsed since an arbitrary reference in time, in milliseconds. The time reference can be any (the start of execution of the program, the UNIX epoch, whatwever) as long as it does not change during program execution.

Returns

A 32 bit integer containing representing the number of milliseconds elapsed since the time reference.

4.3.2.2 uint32_t lh_get_UTC_unix_time ( )

Returns a 32 bit integer with the UNIX time in seconds.

This method must take into account the calls to lh_update_time_offset so that the time it returns is synchronized with the server time. The recommended way of doing this is by storing the time offset between local time and server time, and substracting that difference from local time when this function is called. Look at the implementation of the Linux version of this C Library to see an example of how to do it.

If UNIX time is not available, the seconds elapsed since a constant reference time in the past must be returned. This time reference must not change during the execution of the program, although it may change between device reboots.

Returns

4.3.2.3 void lh_sleep (uint32_t *millis*)

Stops the execution of the program during a given period of time.

Parameters

millis	The number of milliseconds execution will be stopped.

4.3.2.4 void lh_update_time_offset (uint32_t *server_time*)

Updates an internal time reference so that the values returned by lh_get_UTC_unix_time are synchronized with the time of the server.

This function is used internally by the library to synchronize system clock with the server clock.

Parameters

server_time	The server time in seconds.

4.4 abstraction/udp-comm/udp_api.h File Reference

This header file contains all the functions that abstract the UDP communications of the library.

```
#include <stdint.h>
#include "../../core/stun-messaging/stun_message.h"
```

Macros

- #define **LHINGS_SERVER_HOST** "www.lhings.com"
- #define **LHINGS_SERVER_UDP_PORT** "3478"
- #define **MAXBUFLEN** 2048

Functions

- int lh_send_to_server (StunMessage *message)

  Sends to Lhings server (www.lhings.com) UDP port 3478 the bytes that make up the given STUN message.
- uint8_t * lh_receive_from_server (uint16_t *bytes_recv)

  Listens asynchronously to UDP packets sent from Lhings server UDP port 3478.

4.4.1 Detailed Description

This header file contains all the functions that abstract the UDP communications of the library. All the function in this header file belong to the abstraction API of the library and need to be reimplemented when changing platform. The documentation of each function contains all the information about its expected behaviour. This information must be carefully followed when porting the library to other platforms.

4.4.2 Function Documentation

4.4.2.1 uint8_t* lh_receive_from_server (uint16_t * *bytes_recv*)

Listens asynchronously to UDP packets sent from Lhings server UDP port 3478.

The call must not block indefinitely until a packet is received, since this would stall the main loop of execution (this C library is implemented using a single thread). Listening must be done in the same port as the one used by lh_send_to_server.

Parameters

bytes_recv	If not null, the number of bytes received will be stored in the address it points to.

Returns

null if no data is received, otherwise an array of bytes containing the data received.

4.4.2.2 int lh_send_to_server (StunMessage * *message*)

Sends to Lhings server (www.lhings.com) UDP port 3478 the bytes that make up the given STUN message.

Implementations must bind to a port and use it for the entire session, so that the server knows where to send replies and requests to the device.

Parameters

message	

Returns

1 on success, 0 otherwise.

4.5 core/lhings.h File Reference

Main header file for the Lhings C library.

```
#include "utils/data_structures.h"
```

Data Structures

- struct _lh_config
- struct _device_info

  LH_Info contains generic information about the device.
- struct _status_component

  LH_Component is used to define Lhings `typed parameters`*.*
- struct _device

  LH_Device is the main data type used to store the data of a device and its capabilities.
- struct _event

  LH_Event stores the details of an event, like its name and its components, if any.
- struct _action

  LH_Action stores the details of an action, like its name and arguments.

Macros

- #define **MAX_DELAY_BETWEEN_RETRIES_SECS** 120
- #define **DELAY_BETWEEN_KEEPALIVES_SECS** 30
- #define **LOG_DESCRIPTOR** 0
- #define **MAX_STR_TYPE_LEN** 9
- #define **MIN_COMP_JSON_LEN** 22
- #define **MIN_ACTION_JSON_LEN** 40
- #define **MIN_EVENT_JSON_LEN** 27
- #define **MIN_DESCRIPTOR_JSON_LEN** 153

Typedefs

- typedef struct _lh_config **LH_Config**
- typedef struct _device_info LH_Info

  LH_Info contains generic information about the device.
- typedef struct _status_component LH_Component

  LH_Component is used to define Lhings `typed parameters`*.*
- typedef struct _device LH_Device

  LH_Device is the main data type used to store the data of a device and its capabilities.
- typedef struct _event LH_Event

  LH_Event stores the details of an event, like its name and its components, if any.
- typedef struct _action LH_Action

  LH_Action stores the details of an action, like its name and arguments.

Enumerations

- enum LH_ComponentType {
  LH_TYPE_INTEGER, **LH_TYPE_FLOAT**, **LH_TYPE_STRING**, **LH_TYPE_TIMESTAMP**,
  LH_TYPE_BOOLEAN, **LH_TYPE_NO_TYPE** }

  enum reflecting the `Lhings data types`.

Functions

- void setup ()

  This function must be used to define the actions, events and status components of the device.
- void loop ()

  Main loop of execution.
- int lh_start_device (LH_Device *device, char *device_name, char *username, char *password)

  This function starts the execution of the device.
- void lh_set_loop_frequency_hz (double freq)

  Set the frequency at which the function loop will be called.
- void lh_set_loop_frequency_secs (uint32_t secs)

  Set the frequency at which the function loop will be called.
- LH_Component * lh_model_create_component (char *name, LH_ComponentType type, void *value)

  Used to create components, either to define device capabilities (in the function setup) or to define the `payload` *to be sent with an event.*
- void lh_model_free_component (LH_Component *component)

  Frees the memory allocated by lh_model_create_component.
- void lh_model_add_event (LH_Device *device, char *name, LH_List *components)

  Tells the library that the device is able to send the named event.
- void lh_model_add_status_component (LH_Device *device, char *name, LH_ComponentType type, void *value)

  Tells the library that the device has the named status component.
- void lh_model_add_action (LH_Device *device, char *name, char *description, LH_List *arguments, void(*action_function)(LH_Dict *argument_values))

  Tells the library that the device can perform the named action.
- int lh_send_event (LH_Device *device, char *event_name, char *payload, LH_List *components)

  Notifies Lhings about the occurence of the named event, optionally with a `payload`.
- int lh_store_status (LH_Device *device)

  Automatically stores the status of the device in Lhings, using the `Data API`.

Variables

- LH_Device this_device

  This instance of LH_Device is used internally by the library to track the state and configuration of the device while it is being executed.
- LH_Config **config**

4.5.1 Detailed Description

Main header file for the Lhings C library. This header file contains all the main functions used to implement a device that connects to Lhings.

The structure of a program that uses the library is very simple, it has to implement the functions setup() and loop() and, of course, a main() function.

setup() is used to configure the device and is where all the initialization code should go. Configuring the device means telling the Lhings library which are its actions, events and status components (see http://support.lhings.com/Getting-started.html to get a quick introduction into the main Lhings concepts).

Actions, events and status components are added to the Lhings device using the functions lh_model_add_action(), lh_model_add_event() and lh_model_add_status_component() respectively. This functions populate the struct LH_Device this_device adding to it all the information needed so that the Lhings library is able to generate the descriptor of the device (see http://support.lhings.com/The-Device-Descriptor.html for more details about the device descriptor).

The loop() function will be periodically executed at a rate which can be changed calling lh_set_loop_frequency_hz() or lh_set_loop_frequency_secs(). You can use this method to implement anything that your device needs to perform or check periodically.

In addition to this, you can define action functions, which will be executed any time the corresponding action is requested from the Lhings mobile or web apps, or using the Lhings REST API (http://support.lhings.com/Lhings-API-Documentation.html). These action functions should have the following signature:

```
void action_function_example(LH_Dict *function_arguments)
```

A pointer to this functions must be passed to lh_model_add_action() so that the Lhings library knows which function to call when the named action is requested.

Likewise, you can use variables in your code to define status components of your device, passing a pointer to them in the call to the function lh_model_add_status_component(). For instance, if your device is a sensor that measures temperature and you store its value in a `float` variable called `temp`, you can expose its value as a status component passing a pointer to temp to the aforementioned function. Then its real time value could be checked from the mobile and web apps of Lhings and also using the Lhings REST API.

Whenever you need to send an event you can use the function lh_send_event().

In order to start your device, you have to call the function lh_start_device() from your main() function.

Finally a note on conventions used by the library:

- All library API functions start with `lh_*` . In the same way, all library defined data types start with `LH_*` .
- The library makes extensive use of dynamic memory allocation. Some functions return pointers. By convention, this functions will always return a pointer to a dynamically allocated memory block. In the case of primitive data types (`int` , `float`, `char`, etc) the memory must be freed with free() when it is no longer needed. In the case of library defined data types (`LH_Device`, `LH_Action`, `LH_Dict`, `LH_List`, etc.) always exists a function `lh_free_*` that must be used to properly release the memory allocated by those methods. The library itself is checked against memory leaks using `Valgrind` before each release.

See Also

http://support.lhings.com

4.5.2 Typedef Documentation

4.5.2.1 typedef struct _status_component LH_Component

LH_Component is used to define Lhings `typed parameters`.

Typed parameters are used in 3 places in the Lhings library:

- To define status components of the device in the call to lh_model_add_status_component().
- To define event components of the device in the call to lh_model_add_event().
- To define arguments of the actions in the call to lh_model_add_action().

4.5.2.2 typedef struct _device LH_Device

LH_Device is the main data type used to store the data of a device and its capabilities.

It also stores the information about the Lhings account to which the device belongs.

4.5.2.3 typedef struct _device_info LH_Info

LH_Info contains generic information about the device.

This information will be added to the device descriptor.

4.5.3 Function Documentation

4.5.3.1 void lh_model_add_action (LH_Device * *device*, char * *name*, char * *description*, LH_List * *arguments*, void(*)(LH_Dict *argument_values) *action_function*)

Tells the library that the device can perform the named action.

This information will be used by the library to automatically build the device descriptor and send it to Lhings. A device can have as many actions as needed. See http://support.lhings.com/The-Device--Descriptor.html for more details on the device descriptor.

Parameters

device	The device that is capable of performing this action (usually this will be the variable this_device).
name	A string with the name of the action.
description	A string with the description of the action (optional, can be null).
arguments	The list of arguments the action needs to know to be executed. If null is passed no arguments are added.
action_function	A function pointer to the function that actually performs the requested action.

4.5.3.2 void lh_model_add_event (LH_Device * *device*, char * *name*, LH_List * *components*)

Tells the library that the device is able to send the named event.

This information will be used by the library to automatically build the device descriptor and send it to Lhings. A device can send as many events as needed. See http://support.lhings.com/The-Device--Descriptor.html for more details on the device descriptor.

Parameters

device	The device that is able to send the event (usually this will be the variable this_device).
name	A string with the name of the event.
components	A list with the components of the event. If null is passed no components are added. For more details, see http://support.lhings.com/Event-Payload.html#-How-to-use-structured-payloads

4.5.3.3 void lh_model_add_status_component (LH_Device * *device*, char * *name*, LH_ComponentType *type*, void * *value*)

Tells the library that the device has the named status component.

This information will be used by the library to automatically build the device descriptor and send it to Lhings. A device can have up to 8 status components. See http://support.lhings.com/The-Device--Descriptor.html for more details on the device descriptor.

Parameters

device	The device to which the status component belongs (usually this will be the variable this_device).
name	A string with the name of the status component.
type	The type of the status component (see http://support.lhings.com/Typed--Parameters.html for more details).
value	A pointer to the variable that stores the value of the status component.

4.5.3.4 LH_Component* lh_model_create_component (char * *name*, LH_ComponentType *type*, void * *value*)

Used to create components, either to define device capabilities (in the function setup) or to define the payload to be sent with an event.

Parameters

name	The name of the component.
type	The type of the component (see http://support.lhings.com/Typed--Parameters.html for more details).
value	A pointer to the variable that holds the value of the component, if any. If this component has no associated value, pass null.

Returns

A pointer to the LH_Component created. The returned pointer must be freed when it is no longer needed using lh_model_free_component.

4.5.3.5 void lh_model_free_component (LH_Component * *component*)

Frees the memory allocated by lh_model_create_component.

Parameters

component	

4.5.3.6 int lh_send_event (LH_Device * *device*, char * *event_name*, char * *payload*, LH_List * *components*)

Notifies Lhings about the occurence of the named event, optionally with a payload.

Parameters

device	The device that notifies about the occurence of the event (usually this will be the variable this_device).
event_name	A string containing the name of the event.
payload	A string containing the payload of the event (optional, can be null). If both payload and components are not null, then components is discarded and has no effect.
components	A list of LH_Component structs that contains the components that want to be sent with the event (optional, can be null).

Returns

1 if event is sent successfully, 0 otherwise.

4.5.3.7 void lh_set_loop_frequency_hz (double *freq*)

Set the frequency at which the function loop will be called.

The library will do this on a best effort basis, there is no guarantee on the precision of timing between calls.

Parameters

freq	The frequency in hertz at which loop function will be called.

4.5.3.8 void lh_set_loop_frequency_secs (uint32_t *secs*)

Set the frequency at which the function loop will be called.

The library will do this on a best effort basis, there is no guarantee on the precision of timing between calls.

Parameters

secs	The time in seconds between two consecutive calls to loop function.

4.5.3.9 int lh_start_device (LH_Device * *device*, char * *device_name*, char * *username*, char * *password*)

This function starts the execution of the device.

It will connect to Lhings, register the device in Lhings if needed, call setup(), send the descriptor and periodically execute the function loop(). If there are no errors, the call to this function never returns.

Parameters

device	A pointer to a structure LH_Device. In practice, a pointer to the variable this_device (defined in lhings.h) must always be passed.
device_name	A string with the name given to the device.
username	A string with the username of the account of Lhings in which the device will be registered.
password	A string with the password of the account of Lhings in which the device will be registered.

Returns

0 if there is no error. On success this function never returns.

4.5.3.10 int lh_store_status (LH_Device * *device*)

Automatically stores the status of the device in Lhings, using the `Data API`.

Parameters

device	

Returns

4.5.3.11 void loop ( )

Main loop of execution.

This function will be called periodically by the library. The period between two successive calls can be defined using the functions lh_set_loop_frequency_hz and lh_set_loop_frequency_secs

4.5.3.12 void setup ( )

This function must be used to define the actions, events and status components of the device.

In addition all initialization required by code external to the library can be put here.

4.5.4 Variable Documentation

4.5.4.1 LH_Device this_device

This instance of LH_Device is used internally by the library to track the state and configuration of the device while it is being executed.

Its reference is the one that must be passed in the call to lh_start_device().

Index

_action, 2
_device, 2
_device_info, 3
_event, 3
_lh_config, 4
_status_component, 4

abstraction/http-comm/http_api.h, 6
abstraction/permanent-storage/storage_api.h, 8
abstraction/timing/lhings_time.h, 9
abstraction/udp-comm/udp_api.h, 11

collection_item, 4
core/lhings.h, 12

dictionary, 5
dictionary_item, 5

http_api.h
    lh_http_execute_get, 7
    lh_http_execute_post, 7
    lh_http_execute_put, 8
    lh_http_free, 8
http_response, 5

LH_Component
    lhings.h, 14
LH_Device
    lhings.h, 14
LH_Info
    lhings.h, 15
lh_get_UTC_unix_time
    lhings_time.h, 10
lh_get_absolute_time_millis
    lhings_time.h, 10
lh_http_execute_get
    http_api.h, 7
lh_http_execute_post
    http_api.h, 7
lh_http_execute_put
    http_api.h, 8
lh_http_free
    http_api.h, 8
lh_model_add_action
    lhings.h, 15
lh_model_add_event
    lhings.h, 15
lh_model_add_status_component
    lhings.h, 15
lh_model_create_component
    lhings.h, 16
lh_model_free_component
    lhings.h, 16
lh_receive_from_server
    udp_api.h, 11
lh_send_event
    lhings.h, 16
lh_send_to_server
    udp_api.h, 11
lh_set_loop_frequency_hz
    lhings.h, 16
lh_set_loop_frequency_secs
    lhings.h, 17
lh_sleep
    lhings_time.h, 10
lh_start_device
    lhings.h, 17
lh_storage_get_uuid
    storage_api.h, 9
lh_storage_save_uuid
    storage_api.h, 9
lh_store_status
    lhings.h, 17
lh_update_time_offset
    lhings_time.h, 10
lhings.h
    LH_Component, 14
    LH_Device, 14
    LH_Info, 15
    lh_model_add_action, 15
    lh_model_add_event, 15
    lh_model_add_status_component, 15
    lh_model_create_component, 16
    lh_model_free_component, 16
    lh_send_event, 16
    lh_set_loop_frequency_hz, 16
    lh_set_loop_frequency_secs, 17
    lh_start_device, 17
    lh_store_status, 17
    loop, 17
    setup, 17
    this_device, 17
lhings_time.h
    lh_get_UTC_unix_time, 10
    lh_get_absolute_time_millis, 10
    lh_sleep, 10
    lh_update_time_offset, 10
list, 5
loop
    lhings.h, 17

setup
    lhings.h, 17
storage_api.h
    lh_storage_get_uuid, 9
    lh_storage_save_uuid, 9
string, 6
StunAttribute, 6
StunMessage, 6

this_device
    lhings.h, 17

udp_api.h
    lh_receive_from_server, 11
    lh_send_to_server, 11