

DomotlcApp

0.1

Generated by Doxygen 1.8.11

Contents

1	Main page	1
1.1	Introduction	1
1.2	Génération de la dll	1
1.2.1	Step 1: ouvrir le projet	1
1.2.2	Step 2: si windows, ajouter la librairie libwsck32.a au projet	1
1.2.3	Step 3: run pour générer la dll	1
1.3	Utilisation de la dll	1
1.3.1	Step 1: ouvrir un projet	1
1.3.2	Step 2: si windows, ajouter la librairie libwsck32.a au projet	1
1.3.3	Step 2: ajouter le RequeteHttp.lib généré avec la dll dans le projet	1
1.3.4	Step 3: importer requeteHttp.h de ce projet	1
2	Namespace Index	3
2.1	Namespace List	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7

5 Namespace Documentation	9
5.1 happyhttp Namespace Reference	9
5.1.1 Detailed Description	10
5.1.2 Typedef Documentation	10
5.1.2.1 ResponseBegin_CB	10
5.1.2.2 ResponseComplete_CB	10
5.1.2.3 ResponseData_CB	10
5.1.3 Enumeration Type Documentation	10
5.1.3.1 anonymous enum	10
5.1.4 Function Documentation	11
5.1.4.1 atoaddr(const char *address)	11
5.1.4.2 BailOnSocketError(const char *context)	11
5.1.4.3 datawaiting(int sock)	11
6 Class Documentation	13
6.1 happyhttp::Connection Class Reference	13
6.1.1 Constructor & Destructor Documentation	14
6.1.1.1 Connection(const char *host, int port)	14
6.1.1.2 ~Connection()	14
6.1.2 Member Function Documentation	14
6.1.2.1 close()	14
6.1.2.2 connect()	14
6.1.2.3 endheaders()	14
6.1.2.4 outstanding() const	14
6.1.2.5 pump()	14
6.1.2.6 putheader(const char *header, const char *value)	14
6.1.2.7 putheader(const char *header, int numericvalue)	14
6.1.2.8 putrequest(const char *method, const char *url)	14
6.1.2.9 request(const char *method, const char *url, const char *headers[]=0, const unsigned char *body=0, int bodysize=0)	14
6.1.2.10 send(const unsigned char *buf, int numbytes)	14

6.1.2.11	setcallbacks(ResponseBegin_CB begincb, ResponseData_CB datacb, ResponseComplete_CB completecb, void *userdata)	14
6.1.3	Friends And Related Function Documentation	14
6.1.3.1	Response	14
6.1.4	Member Data Documentation	14
6.1.4.1	m_ResponseBeginCB	14
6.1.4.2	m_ResponseCompleteCB	14
6.1.4.3	m_ResponseDataCB	14
6.1.4.4	m_UserData	14
6.2	Requete Class Reference	15
6.2.1	Detailed Description	15
6.2.2	Constructor & Destructor Documentation	15
6.2.2.1	Requete(char *)	15
6.2.2.2	Requete()	15
6.2.2.3	~Requete()	15
6.2.3	Member Function Documentation	15
6.2.3.1	envoyerRequete(char *s1, char *s2)	15
6.2.3.2	Getadresse()	15
6.2.3.3	Setadresse(char *val)	15
6.3	happyhttp::Response Class Reference	16
6.3.1	Constructor & Destructor Documentation	16
6.3.1.1	Response(const char *method, Connection &conn)	16
6.3.2	Member Function Documentation	16
6.3.2.1	completed() const	16
6.3.2.2	getheader(const char *name) const	16
6.3.2.3	getreason() const	16
6.3.2.4	getstatus() const	16
6.3.2.5	notifyconnectionclosed()	16
6.3.2.6	pump(const unsigned char *data, int datasize)	16
6.3.2.7	willclose() const	16
6.3.3	Friends And Related Function Documentation	16
6.3.3.1	Connection	16
6.4	happyhttp::Wobbly Class Reference	17
6.4.1	Member Enumeration Documentation	17
6.4.1.1	anonymous enum	17
6.4.2	Constructor & Destructor Documentation	17
6.4.2.1	Wobbly(const char *fmt,...)	17
6.4.3	Member Function Documentation	17
6.4.3.1	what() const	17
6.4.4	Member Data Documentation	17
6.4.4.1	m_Message	17

7	File Documentation	19
7.1	dll/base64.cpp File Reference	19
7.1.1	Function Documentation	19
7.1.1.1	base64_decode(std::string const &encoded_string)	19
7.1.1.2	base64_encode(unsigned char const *bytes_to_encode, unsigned int in_len)	19
7.2	dll/base64.h File Reference	20
7.2.1	Detailed Description	20
7.2.2	Function Documentation	20
7.2.2.1	base64_decode(std::string const &s)	20
7.2.2.2	base64_encode(unsigned char const *, unsigned int len)	20
7.3	dll/happyhttp.cpp File Reference	21
7.3.1	Macro Definition Documentation	22
7.3.1.1	_stricmp	22
7.4	dll/happyhttp.h File Reference	22
7.4.1	Detailed Description	23
7.5	dll/main.cpp File Reference	23
7.5.1	Function Documentation	23
7.5.1.1	main(int argc, char *argv[])	23
7.6	dll/main.h File Reference	24
7.6.1	Macro Definition Documentation	24
7.6.1.1	DLL_EXPORT	24
7.6.2	Function Documentation	24
7.6.2.1	SomeFunction(const LPCSTR sometext)	24
7.7	dll/Requete.cpp File Reference	24
7.7.1	Function Documentation	25
7.7.1.1	OnBegin(const happyhttp::Response *r, void *userdata)	25
7.7.1.2	OnComplete(const happyhttp::Response *r, void *userdata)	25
7.7.1.3	OnData(const happyhttp::Response *r, void *userdata, const unsigned char *data, int n)	25
7.7.2	Variable Documentation	25
7.7.2.1	count	25

7.8	dll/Requete.h File Reference	25
7.8.1	Detailed Description	25
7.8.2	Macro Definition Documentation	26
7.8.2.1	REQUETE_H	26
7.9	dll/RequeteHttp.cpp File Reference	26
7.9.1	Macro Definition Documentation	26
7.9.1.1	DllExport	26
7.9.2	Function Documentation	26
7.9.2.1	requeteHttpFibaro(char *s1, char *s2, char *user, char *pass)	26
7.9.2.2	requeteHttpKira(char *s1, char *s2)	26
7.10	dll/RequeteHttp.h File Reference	27
7.10.1	Detailed Description	27
7.10.2	Macro Definition Documentation	27
7.10.2.1	DllImport	27
7.10.3	Function Documentation	27
7.10.3.1	requeteHttpFibaro(char *s1, char *s2, char *user, char *pass)	27
7.10.3.2	requeteHttpKira(char *s1, char *s2)	28
	Index	29

Chapter 1

Main page

1.1 Introduction

Ceci est la documentation de la DLL utilisée lors du projet MyDomotik en lien avec l'INSA et Ergovie réalisée par Corentin Chatellier. Cette DLL est directement importée et utilisée dans le modèle de l'application

1.2 Génération de la dll

1.2.1 Step 1: ouvrir le projet

1.2.2 Step 2: si windows, ajouter la librairie libwsck32.a au projet

1.2.3 Step 3: run pour générer la dll

1.3 Utilisation de la dll

1.3.1 Step 1: ouvrir un projet

1.3.2 Step 2: si windows, ajouter la librairie libwsck32.a au projet

1.3.3 Step 2: ajouter le RequeteHttp.lib généré avec la dll dans le projet

1.3.4 Step 3: importer requeteHttp.h de ce projet

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

[happyhttp](#)

Faire des requêtes http, voir <https://github.com/Zintinio/HappyHTTP> 9

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

happyhttp::Connection	13
Requete	
Classe a utilisée par la fonction void DllImport requeteHttpKira(char* s1, char* s2);	15
happyhttp::Response	16
happyhttp::Wobbly	17

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

dll/ base64.cpp	19
dll/ base64.h	
Encoder une chaine de caractères en base64 ou decoder du base64 vers une chaine de caractères	20
dll/ happyhttp.cpp	21
dll/ happyhttp.h	
Faire des requêtes http, voir https://github.com/Zintinio/HappyHTTP	22
dll/ main.cpp	23
dll/ main.h	24
dll/ Requete.cpp	24
dll/ Requete.h	
Classe Requete	25
dll/ RequeteHttp.cpp	26
dll/ RequeteHttp.h	
Fonctions utilisables avec la DLL	27

Chapter 5

Namespace Documentation

5.1 happyhttp Namespace Reference

faire des requêtes http, voir <https://github.com/Zintinio/HappyHTTP>

Classes

- class [Connection](#)
- class [Response](#)
- class [Wobbly](#)

Typedefs

- typedef void(* [ResponseBegin_CB](#)) (const [Response](#) *r, void *userdata)
- typedef void(* [ResponseData_CB](#)) (const [Response](#) *r, void *userdata, const unsigned char *data, int num-bytes)
- typedef void(* [ResponseComplete_CB](#)) (const [Response](#) *r, void *userdata)

Enumerations

- enum {
 [CONTINUE](#) = 100, [SWITCHING_PROTOCOLS](#) = 101, [PROCESSING](#) = 102, [OK](#) = 200,
 [CREATED](#) = 201, [ACCEPTED](#) = 202, [NON_AUTHORITATIVE_INFORMATION](#) = 203, [NO_CONTENT](#) =
 204,
 [RESET_CONTENT](#) = 205, [PARTIAL_CONTENT](#) = 206, [MULTI_STATUS](#) = 207, [IM_USED](#) = 226,
 [MULTIPLE_CHOICES](#) = 300, [MOVED_PERMANENTLY](#) = 301, [FOUND](#) = 302, [SEE_OTHER](#) = 303,
 [NOT_MODIFIED](#) = 304, [USE_PROXY](#) = 305, [TEMPORARY_REDIRECT](#) = 307, [BAD_REQUEST](#) = 400,
 [UNAUTHORIZED](#) = 401, [PAYMENT_REQUIRED](#) = 402, [FORBIDDEN](#) = 403, [NOT_FOUND](#) = 404,
 [METHOD_NOT_ALLOWED](#) = 405, [NOT_ACCEPTABLE](#) = 406, [PROXY_AUTHENTICATION_REQUIRED](#) =
 407, [REQUEST_TIMEOUT](#) = 408,
 [CONFLICT](#) = 409, [GONE](#) = 410, [LENGTH_REQUIRED](#) = 411, [PRECONDITION_FAILED](#) = 412,
 [REQUEST_ENTITY_TOO_LARGE](#) = 413, [REQUEST_URI_TOO_LONG](#) = 414, [UNSUPPORTED_MEDIA_↵](#)
 [_TYPE](#) = 415, [REQUESTED_RANGE_NOT_SATISFIABLE](#) = 416,
 [EXPECTATION_FAILED](#) = 417, [UNPROCESSABLE_ENTITY](#) = 422, [LOCKED](#) = 423, [FAILED_DEPEND_↵](#)
 [ENCY](#) = 424,
 [UPGRADE_REQUIRED](#) = 426, [INTERNAL_SERVER_ERROR](#) = 500, [NOT_IMPLEMENTED](#) = 501, [BAD_↵](#)
 [_GATEWAY](#) = 502,
 [SERVICE_UNAVAILABLE](#) = 503, [GATEWAY_TIMEOUT](#) = 504, [HTTP_VERSION_NOT_SUPPORTED](#) =
 505, [INSUFFICIENT_STORAGE](#) = 507,
 [NOT_EXTENDED](#) = 510 }

Functions

- void [BailOnSocketError](#) (const char *context)
- bool [dataawaiting](#) (int sock)
- struct in_addr * [atoaddr](#) (const char *address)

5.1.1 Detailed Description

faire des requêtes http, voir <https://github.com/Zintinio/HappyHTTP>

5.1.2 Typedef Documentation

5.1.2.1 typedef void(* happyhttp::ResponseBegin_CB) (const Response *r, void *userdata)

5.1.2.2 typedef void(* happyhttp::ResponseComplete_CB) (const Response *r, void *userdata)

5.1.2.3 typedef void(* happyhttp::ResponseData_CB) (const Response *r, void *userdata, const unsigned char *data, int numbytes)

5.1.3 Enumeration Type Documentation

5.1.3.1 anonymous enum

Enumerator

CONTINUE
SWITCHING_PROTOCOLS
PROCESSING
OK
CREATED
ACCEPTED
NON_AUTHORITATIVE_INFORMATION
NO_CONTENT
RESET_CONTENT
PARTIAL_CONTENT
MULTI_STATUS
IM_USED
MULTIPLE_CHOICES
MOVED_PERMANENTLY
FOUND
SEE_OTHER
NOT_MODIFIED
USE_PROXY
TEMPORARY_REDIRECT
BAD_REQUEST
UNAUTHORIZED

PAYMENT_REQUIRED
FORBIDDEN
NOT_FOUND
METHOD_NOT_ALLOWED
NOT_ACCEPTABLE
PROXY_AUTHENTICATION_REQUIRED
REQUEST_TIMEOUT
CONFLICT
GONE
LENGTH_REQUIRED
PRECONDITION_FAILED
REQUEST_ENTITY_TOO_LARGE
REQUEST_URI_TOO_LONG
UNSUPPORTED_MEDIA_TYPE
REQUESTED_RANGE_NOT_SATISFIABLE
EXPECTATION_FAILED
UNPROCESSABLE_ENTITY
LOCKED
FAILED_DEPENDENCY
UPGRADE_REQUIRED
INTERNAL_SERVER_ERROR
NOT_IMPLEMENTED
BAD_GATEWAY
SERVICE_UNAVAILABLE
GATEWAY_TIMEOUT
HTTP_VERSION_NOT_SUPPORTED
INSUFFICIENT_STORAGE
NOT_EXTENDED

5.1.4 Function Documentation

5.1.4.1 `struct in_addr * happyhttp::atoaddr (const char * address)`

5.1.4.2 `void happyhttp::BailOnSocketError (const char * context)`

5.1.4.3 `bool happyhttp::datawaiting (int sock)`

Chapter 6

Class Documentation

6.1 happyhttp::Connection Class Reference

```
#include <happyhttp.h>
```

Public Member Functions

- [Connection](#) (const char *host, int port)
- [~Connection](#) ()
- void [setcallbacks](#) ([ResponseBegin_CB](#) begincb, [ResponseData_CB](#) datacb, [ResponseComplete_CB](#) completecb, void *userdata)
- void [connect](#) ()
- void [close](#) ()
- void [pump](#) ()
- bool [outstanding](#) () const
- void [request](#) (const char *method, const char *url, const char *headers[]=0, const unsigned char *body=0, int bodysize=0)
- void [putrequest](#) (const char *method, const char *url)
- void [putheader](#) (const char *header, const char *value)
- void [putheader](#) (const char *header, int numericvalue)
- void [endheaders](#) ()
- void [send](#) (const unsigned char *buf, int numbytes)

Protected Attributes

- [ResponseBegin_CB](#) m_ResponseBeginCB
- [ResponseData_CB](#) m_ResponseDataCB
- [ResponseComplete_CB](#) m_ResponseCompleteCB
- void * [m_UserData](#)

Friends

- class [Response](#)

6.1.1 Constructor & Destructor Documentation

6.1.1.1 `happyhttp::Connection::Connection (const char * host, int port)`

6.1.1.2 `happyhttp::Connection::~~Connection ()`

6.1.2 Member Function Documentation

6.1.2.1 `void happyhttp::Connection::close ()`

6.1.2.2 `void happyhttp::Connection::connect ()`

6.1.2.3 `void happyhttp::Connection::endheaders ()`

6.1.2.4 `bool happyhttp::Connection::outstanding () const [inline]`

6.1.2.5 `void happyhttp::Connection::pump ()`

6.1.2.6 `void happyhttp::Connection::putheader (const char * header, const char * value)`

6.1.2.7 `void happyhttp::Connection::putheader (const char * header, int numericvalue)`

6.1.2.8 `void happyhttp::Connection::putrequest (const char * method, const char * url)`

6.1.2.9 `void happyhttp::Connection::request (const char * method, const char * url, const char * headers[] = 0, const unsigned char * body = 0, int bodysize = 0)`

6.1.2.10 `void happyhttp::Connection::send (const unsigned char * buf, int numbytes)`

6.1.2.11 `void happyhttp::Connection::setcallbacks (ResponseBegin_CB begincb, ResponseData_CB datacb, ResponseComplete_CB completecb, void * userdata)`

6.1.3 Friends And Related Function Documentation

6.1.3.1 `friend class Response [friend]`

6.1.4 Member Data Documentation

6.1.4.1 `ResponseBegin_CB happyhttp::Connection::m_ResponseBeginCB [protected]`

6.1.4.2 `ResponseComplete_CB happyhttp::Connection::m_ResponseCompleteCB [protected]`

6.1.4.3 `ResponseData_CB happyhttp::Connection::m_ResponseDataCB [protected]`

6.1.4.4 `void* happyhttp::Connection::m_UserData [protected]`

The documentation for this class was generated from the following files:

- [dll/happyhttp.h](#)
- [dll/happyhttp.cpp](#)

6.2 Requete Class Reference

classe a utilisée par la fonction void DllImport [requeteHttpKira\(char* s1, char* s2\);](#)

```
#include <Requete.h>
```

Public Member Functions

- [Requete](#) (char *)
- [Requete](#) ()
- virtual [~Requete](#) ()
- char * [Getadresse](#) ()
- void [Setadresse](#) (char *val)
- int [envoyerRequete](#) (char *s1, char *s2)

6.2.1 Detailed Description

classe a utilisée par la fonction void DllImport [requeteHttpKira\(char* s1, char* s2\);](#)

La classe utilise Happyhttp pour faire une requête

6.2.2 Constructor & Destructor Documentation

6.2.2.1 [Requete::Requete \(char * s1 \)](#)

6.2.2.2 [Requete::Requete \(\)](#)

6.2.2.3 [Requete::~~Requete \(\)](#) [virtual]

6.2.3 Member Function Documentation

6.2.3.1 [int Requete::envoyerRequete \(char * s1, char * s2 \)](#)

Parameters

in	char*	s1 IP
in	char*	s2 reste de l'adresse

6.2.3.2 [char* Requete::Getadresse \(\)](#) [inline]

6.2.3.3 [void Requete::Setadresse \(char * val \)](#) [inline]

The documentation for this class was generated from the following files:

- [dll/Requete.h](#)
- [dll/Requete.cpp](#)

6.3 happyhttp::Response Class Reference

```
#include <happyhttp.h>
```

Public Member Functions

- const char * [getheader](#) (const char *name) const
- bool [completed](#) () const
- int [getstatus](#) () const
- const char * [getreason](#) () const
- bool [willclose](#) () const

Protected Member Functions

- [Response](#) (const char *method, [Connection](#) &conn)
- int [pump](#) (const unsigned char *data, int datasize)
- void [notifyconnectionclosed](#) ()

Friends

- class [Connection](#)

6.3.1 Constructor & Destructor Documentation

6.3.1.1 `happyhttp::Response::Response (const char * method, Connection & conn)` `[protected]`

6.3.2 Member Function Documentation

6.3.2.1 `bool happyhttp::Response::completed () const` `[inline]`

6.3.2.2 `const char * happyhttp::Response::getheader (const char * name) const`

6.3.2.3 `const char * happyhttp::Response::getreason () const`

6.3.2.4 `int happyhttp::Response::getstatus () const`

6.3.2.5 `void happyhttp::Response::notifyconnectionclosed ()` `[protected]`

6.3.2.6 `int happyhttp::Response::pump (const unsigned char * data, int datasize)` `[protected]`

6.3.2.7 `bool happyhttp::Response::willclose () const` `[inline]`

6.3.3 Friends And Related Function Documentation

6.3.3.1 `friend class Connection` `[friend]`

The documentation for this class was generated from the following files:

- [dll/happyhttp.h](#)
- [dll/happyhttp.cpp](#)

6.4 happyhttp::Wobbly Class Reference

```
#include <happyhttp.h>
```

Public Member Functions

- [Wobbly](#) (const char *fmt,...)
- const char * [what](#) () const

Protected Types

- enum { [MAXLEN](#) =256 }

Protected Attributes

- char [m_Message](#) [[MAXLEN](#)]

6.4.1 Member Enumeration Documentation

6.4.1.1 anonymous enum [protected]

Enumerator

MAXLEN

6.4.2 Constructor & Destructor Documentation

6.4.2.1 happyhttp::Wobbly::Wobbly (const char * *fmt*, ...)

6.4.3 Member Function Documentation

6.4.3.1 const char* happyhttp::Wobbly::what () const [inline]

6.4.4 Member Data Documentation

6.4.4.1 char happyhttp::Wobbly::m_Message[[MAXLEN](#)] [protected]

The documentation for this class was generated from the following files:

- [dll/happyhttp.h](#)
- [dll/happyhttp.cpp](#)

Chapter 7

File Documentation

7.1 dll/base64.cpp File Reference

```
#include "base64.h"
#include <iostream>
```

Functions

- `std::string base64_encode` (unsigned char const *bytes_to_encode, unsigned int in_len)
- `std::string base64_decode` (std::string const &encoded_string)

7.1.1 Function Documentation

7.1.1.1 `std::string base64_decode (std::string const & s)`

Parameters

in	<i>std::string</i>	const& s : la chaine à décoder
----	--------------------	--------------------------------

Returns

string ret : la chaine décodée

7.1.1.2 `string base64_encode (unsigned char const * bytes_to_encode, unsigned int in_len)`

Parameters

in	<i>unsigned</i>	char const* bytes_to_encode : la chaine à encoder
in	<i>unsigned</i>	int in_len : taille de la chaine à encoder

Returns

string ret : la chaine encodée

7.2 dll/base64.h File Reference

encoder une chaine de caractères en base64 ou decoder du base64 vers une chaine de caractères

```
#include <string>
```

Functions

- std::string [base64_encode](#) (unsigned char const *, unsigned int len)
- std::string [base64_decode](#) (std::string const &s)

7.2.1 Detailed Description

encoder une chaine de caractères en base64 ou decoder du base64 vers une chaine de caractères

Author

Corentin Chatellier

Version

0.1

Date

14/05/2016

7.2.2 Function Documentation

7.2.2.1 std::string [base64_decode](#) (std::string const & *encoded_string*)

Parameters

in	<i>std::string</i>	const& s : la chaine à décoder
----	--------------------	--------------------------------

Returns

string ret : la chaine décodée

7.2.2.2 std::string [base64_encode](#) (unsigned char const * *bytes_to_encode*, unsigned int *in_len*)

Parameters

in	<i>unsigned</i>	char const* bytes_to_encode : la chaine à encoder
in	<i>unsigned</i>	int in_len : taille de la chaine à encoder

Returns

string ret : la chaine encodée

7.3 dll/happyhttp.cpp File Reference

```
#include "happyhttp.h"
#include <cstdio>
#include <cstring>
#include <stdlib.h>
#include <stdio.h>
#include <Winsock2.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <errno.h>
#include <unistd.h>
#include <cstdarg>
#include <assert.h>
#include <string>
#include <vector>
#include <algorithm>
```

Namespaces

- [happyhttp](#)

faire des requêtes http, voir <https://github.com/Zintinio/HappyHTTP>

Macros

- `#define _stricmp strcasecmp`

Functions

- void [happyhttp::BailOnSocketError](#) (const char *context)
- bool [happyhttp::datawaiting](#) (int sock)
- struct in_addr * [happyhttp::atoaddr](#) (const char *address)

7.3.1 Macro Definition Documentation

7.3.1.1 `#define _stricmp strcasecmp`

7.4 dll/happyhttp.h File Reference

faire des requêtes http, voir <https://github.com/Zintinio/HappyHTTP>

```
#include <string>
#include <map>
#include <vector>
#include <deque>
```

Classes

- class [happyhttp::Wobbly](#)
- class [happyhttp::Connection](#)
- class [happyhttp::Response](#)

Namespaces

- [happyhttp](#)

faire des requêtes http, voir <https://github.com/Zintinio/HappyHTTP>

Typedefs

- typedef void(* [happyhttp::ResponseBegin_CB](#)) (const Response *r, void *userdata)
- typedef void(* [happyhttp::ResponseData_CB](#)) (const Response *r, void *userdata, const unsigned char *data, int numbytes)
- typedef void(* [happyhttp::ResponseComplete_CB](#)) (const Response *r, void *userdata)

Enumerations

- enum {
[happyhttp::CONTINUE](#) = 100, [happyhttp::SWITCHING_PROTOCOLS](#) = 101, [happyhttp::PROCESSING](#) = 102, [happyhttp::OK](#) = 200,
[happyhttp::CREATED](#) = 201, [happyhttp::ACCEPTED](#) = 202, [happyhttp::NON_AUTHORITATIVE_INFORMATION](#) = 203, [happyhttp::NO_CONTENT](#) = 204,
[happyhttp::RESET_CONTENT](#) = 205, [happyhttp::PARTIAL_CONTENT](#) = 206, [happyhttp::MULTI_STATUS](#) = 207, [happyhttp::IM_USED](#) = 226,
[happyhttp::MULTIPLE_CHOICES](#) = 300, [happyhttp::MOVED_PERMANENTLY](#) = 301, [happyhttp::FOUND](#) = 302, [happyhttp::SEE_OTHER](#) = 303,
[happyhttp::NOT_MODIFIED](#) = 304, [happyhttp::USE_PROXY](#) = 305, [happyhttp::TEMPORARY_REDIRECT](#) = 307, [happyhttp::BAD_REQUEST](#) = 400,
[happyhttp::UNAUTHORIZED](#) = 401, [happyhttp::PAYMENT_REQUIRED](#) = 402, [happyhttp::FORBIDDEN](#) =

```

403, happyhttp::NOT_FOUND = 404,
happyhttp::METHOD_NOT_ALLOWED = 405, happyhttp::NOT_ACCEPTABLE = 406, happyhttp::PROXY_↵
_AUTHENTICATION_REQUIRED = 407, happyhttp::REQUEST_TIMEOUT = 408,
happyhttp::CONFLICT = 409, happyhttp::GONE = 410, happyhttp::LENGTH_REQUIRED = 411, happyhttp_↵
::PRECONDITION_FAILED = 412,
happyhttp::REQUEST_ENTITY_TOO_LARGE = 413, happyhttp::REQUEST_URI_TOO_LONG = 414,
happyhttp::UNSUPPORTED_MEDIA_TYPE = 415, happyhttp::REQUESTED_RANGE_NOT_SATISFI_↵
ABLE = 416,
happyhttp::EXPECTATION_FAILED = 417, happyhttp::UNPROCESSABLE_ENTITY = 422, happyhttp::L_↵
OCKED = 423, happyhttp::FAILED_DEPENDENCY = 424,
happyhttp::UPGRADE_REQUIRED = 426, happyhttp::INTERNAL_SERVER_ERROR = 500, happyhttp::N_↵
OT_IMPLEMENTED = 501, happyhttp::BAD_GATEWAY = 502,
happyhttp::SERVICE_UNAVAILABLE = 503, happyhttp::GATEWAY_TIMEOUT = 504, happyhttp::HTTP_↵
_VERSION_NOT_SUPPORTED = 505, happyhttp::INSUFFICIENT_STORAGE = 507,
happyhttp::NOT_EXTENDED = 510 }

```

Functions

- void `happyhttp::BailOnSocketError` (const char *context)
- struct in_addr * `happyhttp::atoaddr` (const char *address)

7.4.1 Detailed Description

faire des requêtes http, voir <https://github.com/Zintinio/HappyHTTP>

Date

14/05/2016

7.5 dll/main.cpp File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include "RequeteHttp.h"

```

Functions

- int `main` (int argc, char *argv[])

7.5.1 Function Documentation

7.5.1.1 int main (int argc, char * argv[])

exemple

7.6 dll/main.h File Reference

```
#include <windows.h>
```

Macros

- #define [DLL_EXPORT](#) __declspec(dllimport)

Functions

- void [DLL_EXPORT SomeFunction](#) (const LPCSTR sometext)

7.6.1 Macro Definition Documentation

7.6.1.1 #define [DLL_EXPORT](#) __declspec(dllimport)

7.6.2 Function Documentation

7.6.2.1 void [DLL_EXPORT SomeFunction](#) (const LPCSTR *sometext*)

7.7 dll/Requete.cpp File Reference

```
#include "Requete.h"  
#include "happyhttp.h"  
#include <stdio>  
#include <string>
```

Functions

- void [OnBegin](#) (const [happyhttp::Response](#) *r, void *userdata)
- void [OnData](#) (const [happyhttp::Response](#) *r, void *userdata, const unsigned char *data, int n)
- void [OnComplete](#) (const [happyhttp::Response](#) *r, void *userdata)

Variables

- int [count](#) =0

7.7.1 Function Documentation

7.7.1.1 void OnBegin (const happyhttp::Response * *r*, void * *userdata*)

7.7.1.2 void OnComplete (const happyhttp::Response * *r*, void * *userdata*)

7.7.1.3 void OnData (const happyhttp::Response * *r*, void * *userdata*, const unsigned char * *data*, int *n*)

7.7.2 Variable Documentation

7.7.2.1 int count =0

7.8 dll/Requete.h File Reference

classe [Requete](#)

```
#include "happyhttp.h"
#include <cstdio>
#include <cstring>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <wtypes.h>
```

Classes

- class [Requete](#)

classe a utilisée par la fonction void DllImport [requeteHttpKira\(char s1, char* s2\);](#)*

Macros

- #define [REQUETE_H](#)

7.8.1 Detailed Description

classe [Requete](#)

Author

Corentin Chatellier

Version

0.1

Date

14/05/2016

7.8.2 Macro Definition Documentation

7.8.2.1 #define REQUETE_H

7.9 dll/RequeteHttp.cpp File Reference

```
#include <string>
#include "RequeteHttp.h"
#include "happyhttp.h"
#include "Requete.h"
#include "base64.h"
```

Macros

- #define [DllExport](#) __declspec(dllexport)

Functions

- void [DllExport requeteHttpKira](#) (char *s1, char *s2)
- void [DllExport requeteHttpFibaro](#) (char *s1, char *s2, char *user, char *pass)

7.9.1 Macro Definition Documentation

7.9.1.1 #define DllExport __declspec(dllexport)

7.9.2 Function Documentation

7.9.2.1 void DllImport requeteHttpFibaro (char * s1, char * s2, char * user, char * pass)

Parameters

in	<i>char*</i>	s1 : IP
in	<i>char*</i>	s2 : reste de l'adresse
in	<i>char*</i>	user : login de la fibaro
in	<i>char*</i>	pass : mot de passe de la fibaro

Returns

void

7.9.2.2 void DllImport requeteHttpKira (char * s1, char * s2)

Parameters

in	char*	s1 IP
in	char*	s2 reste de l'adresse

Returns

void

7.10 dll/RequeteHttp.h File Reference

fonctions utilisables avec la DLL

Macros

- #define [DllImport](#) __declspec(dllimport)

Functions

- void [DllImport requeteHttpKira](#) (char *s1, char *s2)
- void [DllImport requeteHttpFibaro](#) (char *s1, char *s2, char *user, char *pass)

7.10.1 Detailed Description

fonctions utilisables avec la DLL

Author

Corentin Chatellier

Version

0.1

Date

14/05/2016

7.10.2 Macro Definition Documentation

7.10.2.1 #define [DllImport](#) __declspec(dllimport)

7.10.3 Function Documentation

7.10.3.1 void [DllImport requeteHttpFibaro](#) (char * *s1*, char * *s2*, char * *user*, char * *pass*)

Parameters

in	<i>char*</i>	s1 : IP
in	<i>char*</i>	s2 : reste de l'adresse
in	<i>char*</i>	user : login de la fibaro
in	<i>char*</i>	pass : mot de passe de la fibaro

Returns

void

7.10.3.2 void DllImport requeteHttpKira (char * s1, char * s2)**Parameters**

in	<i>char*</i>	s1 IP
in	<i>char*</i>	s2 reste de l'adresse

Returns

void

Index

- `_stricmp`
 - `happyhttp.cpp`, 22
 - `~Connection`
 - `happyhttp::Connection`, 14
 - `~Requete`
 - `Requete`, 15
- ACCEPTED
 - `happyhttp`, 10
- `atoaddr`
 - `happyhttp`, 11
- BAD_GATEWAY
 - `happyhttp`, 11
- BAD_REQUEST
 - `happyhttp`, 10
- `BailOnSocketError`
 - `happyhttp`, 11
- `base64.cpp`
 - `base64_decode`, 19
 - `base64_encode`, 19
- `base64.h`
 - `base64_decode`, 20
 - `base64_encode`, 20
- `base64_decode`
 - `base64.cpp`, 19
 - `base64.h`, 20
- `base64_encode`
 - `base64.cpp`, 19
 - `base64.h`, 20
- CONFLICT
 - `happyhttp`, 11
- CONTINUE
 - `happyhttp`, 10
- CREATED
 - `happyhttp`, 10
- `close`
 - `happyhttp::Connection`, 14
- `completed`
 - `happyhttp::Response`, 16
- `connect`
 - `happyhttp::Connection`, 14
- `Connection`
 - `happyhttp::Connection`, 14
 - `happyhttp::Response`, 16
- `count`
 - `Requete.cpp`, 25
- DLL_EXPORT
 - `main.h`, 24
- `datawaiting`
 - `happyhttp`, 11
- `dll/Requete.cpp`, 24
- `dll/Requete.h`, 25
- `dll/RequeteHttp.cpp`, 26
- `dll/RequeteHttp.h`, 27
- `dll/base64.cpp`, 19
- `dll/base64.h`, 20
- `dll/happyhttp.cpp`, 21
- `dll/happyhttp.h`, 22
- `dll/main.cpp`, 23
- `dll/main.h`, 24
- DllExport
 - `RequeteHttp.cpp`, 26
- DllImport
 - `RequeteHttp.h`, 27
- EXPECTATION_FAILED
 - `happyhttp`, 11
- `endheaders`
 - `happyhttp::Connection`, 14
- `envoyerRequete`
 - `Requete`, 15
- FAILED_DEPENDENCY
 - `happyhttp`, 11
- FORBIDDEN
 - `happyhttp`, 11
- FOUND
 - `happyhttp`, 10
- GATEWAY_TIMEOUT
 - `happyhttp`, 11
- GONE
 - `happyhttp`, 11
- `Getadresse`
 - `Requete`, 15
- `getheader`
 - `happyhttp::Response`, 16
- `getreason`
 - `happyhttp::Response`, 16
- `getstatus`
 - `happyhttp::Response`, 16
- HTTP_VERSION_NOT_SUPPORTED
 - `happyhttp`, 11
- `happyhttp`, 9
 - ACCEPTED, 10
 - `atoaddr`, 11

- BAD_GATEWAY, 11
- BAD_REQUEST, 10
- BailOnSocketError, 11
- CONFLICT, 11
- CONTINUE, 10
- CREATED, 10
- datawaiting, 11
- EXPECTATION_FAILED, 11
- FAILED_DEPENDENCY, 11
- FORBIDDEN, 11
- FOUND, 10
- GATEWAY_TIMEOUT, 11
- GONE, 11
- HTTP_VERSION_NOT_SUPPORTED, 11
- IM_USED, 10
- INSUFFICIENT_STORAGE, 11
- INTERNAL_SERVER_ERROR, 11
- LENGTH_REQUIRED, 11
- LOCKED, 11
- METHOD_NOT_ALLOWED, 11
- MOVED_PERMANENTLY, 10
- MULTI_STATUS, 10
- MULTIPLE_CHOICES, 10
- NO_CONTENT, 10
- NON_AUTHORITATIVE_INFORMATION, 10
- NOT_ACCEPTABLE, 11
- NOT_EXTENDED, 11
- NOT_FOUND, 11
- NOT_IMPLEMENTED, 11
- NOT_MODIFIED, 10
- OK, 10
- PARTIAL_CONTENT, 10
- PAYMENT_REQUIRED, 10
- PRECONDITION_FAILED, 11
- PROCESSING, 10
- PROXY_AUTHENTICATION_REQUIRED, 11
- REQUEST_ENTITY_TOO_LARGE, 11
- REQUEST_TIMEOUT, 11
- REQUEST_URI_TOO_LONG, 11
- REQUESTED_RANGE_NOT_SATISFIABLE, 11
- RESET_CONTENT, 10
- ResponseBegin_CB, 10
- ResponseComplete_CB, 10
- ResponseData_CB, 10
- SEE_OTHER, 10
- SERVICE_UNAVAILABLE, 11
- SWITCHING_PROTOCOLS, 10
- TEMPORARY_REDIRECT, 10
- UNAUTHORIZED, 10
- UNPROCESSABLE_ENTITY, 11
- UNSUPPORTED_MEDIA_TYPE, 11
- UPGRADE_REQUIRED, 11
- USE_PROXY, 10
- happyhttp.cpp
 - _stricmp, 22
- happyhttp::Connection, 13
 - ~Connection, 14
 - close, 14
 - connect, 14
 - Connection, 14
 - endheaders, 14
 - m_ResponseBeginCB, 14
 - m_ResponseCompleteCB, 14
 - m_ResponseDataCB, 14
 - m_UserData, 14
 - outstanding, 14
 - pump, 14
 - putheader, 14
 - putrequest, 14
 - request, 14
 - Response, 14
 - send, 14
 - setcallbacks, 14
- happyhttp::Response, 16
 - completed, 16
 - Connection, 16
 - getheader, 16
 - getreason, 16
 - getstatus, 16
 - notifyconnectionclosed, 16
 - pump, 16
 - Response, 16
 - willclose, 16
- happyhttp::Wobbly, 17
 - m_Message, 17
 - MAXLEN, 17
 - what, 17
 - Wobbly, 17
- IM_USED
 - happyhttp, 10
- INSUFFICIENT_STORAGE
 - happyhttp, 11
- INTERNAL_SERVER_ERROR
 - happyhttp, 11
- LENGTH_REQUIRED
 - happyhttp, 11
- LOCKED
 - happyhttp, 11
- m_Message
 - happyhttp::Wobbly, 17
- m_ResponseBeginCB
 - happyhttp::Connection, 14
- m_ResponseCompleteCB
 - happyhttp::Connection, 14
- m_ResponseDataCB
 - happyhttp::Connection, 14
- m_UserData
 - happyhttp::Connection, 14
- MAXLEN
 - happyhttp::Wobbly, 17
- METHOD_NOT_ALLOWED
 - happyhttp, 11
- MOVED_PERMANENTLY
 - happyhttp, 10

- MULTI_STATUS
 - happyhttp, [10](#)
- MULTIPLE_CHOICES
 - happyhttp, [10](#)
- main
 - main.cpp, [23](#)
- main.cpp
 - main, [23](#)
- main.h
 - DLL_EXPORT, [24](#)
 - SomeFunction, [24](#)
- NO_CONTENT
 - happyhttp, [10](#)
- NON_AUTHORITATIVE_INFORMATION
 - happyhttp, [10](#)
- NOT_ACCEPTABLE
 - happyhttp, [11](#)
- NOT_EXTENDED
 - happyhttp, [11](#)
- NOT_FOUND
 - happyhttp, [11](#)
- NOT_IMPLEMENTED
 - happyhttp, [11](#)
- NOT_MODIFIED
 - happyhttp, [10](#)
- notifyconnectionclosed
 - happyhttp::Response, [16](#)
- OK
 - happyhttp, [10](#)
- OnBegin
 - Requete.cpp, [25](#)
- OnComplete
 - Requete.cpp, [25](#)
- OnData
 - Requete.cpp, [25](#)
- outstanding
 - happyhttp::Connection, [14](#)
- PARTIAL_CONTENT
 - happyhttp, [10](#)
- PAYMENT_REQUIRED
 - happyhttp, [10](#)
- PRECONDITION_FAILED
 - happyhttp, [11](#)
- PROCESSING
 - happyhttp, [10](#)
- PROXY_AUTHENTICATION_REQUIRED
 - happyhttp, [11](#)
- pump
 - happyhttp::Connection, [14](#)
 - happyhttp::Response, [16](#)
- putheader
 - happyhttp::Connection, [14](#)
- putrequest
 - happyhttp::Connection, [14](#)
- REQUEST_ENTITY_TOO_LARGE
 - happyhttp, [11](#)
- REQUEST_TIMEOUT
 - happyhttp, [11](#)
- REQUEST_URI_TOO_LONG
 - happyhttp, [11](#)
- REQUESTED_RANGE_NOT_SATISFIABLE
 - happyhttp, [11](#)
- REQUETE_H
 - Requete.h, [26](#)
- RESET_CONTENT
 - happyhttp, [10](#)
- request
 - happyhttp::Connection, [14](#)
- Requete, [15](#)
 - ~Requete, [15](#)
 - envoyerRequete, [15](#)
 - Getadresse, [15](#)
 - Requete, [15](#)
 - Setadresse, [15](#)
- Requete.cpp
 - count, [25](#)
 - OnBegin, [25](#)
 - OnComplete, [25](#)
 - OnData, [25](#)
- Requete.h
 - REQUETE_H, [26](#)
- RequeteHttp.cpp
 - DllExport, [26](#)
 - requeteHttpFibaro, [26](#)
 - requeteHttpKira, [26](#)
- RequeteHttp.h
 - DllImport, [27](#)
 - requeteHttpFibaro, [27](#)
 - requeteHttpKira, [28](#)
- requeteHttpFibaro
 - RequeteHttp.cpp, [26](#)
 - RequeteHttp.h, [27](#)
- requeteHttpKira
 - RequeteHttp.cpp, [26](#)
 - RequeteHttp.h, [28](#)
- Response
 - happyhttp::Connection, [14](#)
 - happyhttp::Response, [16](#)
- ResponseBegin_CB
 - happyhttp, [10](#)
- ResponseComplete_CB
 - happyhttp, [10](#)
- ResponseData_CB
 - happyhttp, [10](#)
- SEE_OTHER
 - happyhttp, [10](#)
- SERVICE_UNAVAILABLE
 - happyhttp, [11](#)
- SWITCHING_PROTOCOLS
 - happyhttp, [10](#)
- send
 - happyhttp::Connection, [14](#)
- Setadresse

- Requete, [15](#)
- setcallbacks
 - happyhttp::Connection, [14](#)
- SomeFunction
 - main.h, [24](#)
- TEMPORARY_REDIRECT
 - happyhttp, [10](#)
- UNAUTHORIZED
 - happyhttp, [10](#)
- UNPROCESSABLE_ENTITY
 - happyhttp, [11](#)
- UNSUPPORTED_MEDIA_TYPE
 - happyhttp, [11](#)
- UPGRADE_REQUIRED
 - happyhttp, [11](#)
- USE_PROXY
 - happyhttp, [10](#)
- what
 - happyhttp::Wobbly, [17](#)
- willclose
 - happyhttp::Response, [16](#)
- Wobbly
 - happyhttp::Wobbly, [17](#)