DomotIcApp

0.1

Generated by Doxygen 1.8.11

Contents

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 libwsock32.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 libwsock32.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:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

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 numbytes)
- 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 datawaiting (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	*std::string*	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	*unsigned*	char const* bytes_to_encode : la chaine à encoder
in	*unsigned*	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	*std::string*	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	*unsigned*	char const* bytes_to_encode : la chaine à encoder
in	*unsigned*	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_INFOR↩MATION = 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 <cstdio>
#include <cstring>
```

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	*char**	s1 : IP
in	*char**	s2 : reste de l'adresse
in	*char**	user : login de la fibaro
in	*char**	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	*char∗*	s1 : IP
in	*char∗*	s2 : reste de l'adresse
in	*char∗*	user : login de la fibaro
in	*char∗*	pass : mot de passe de la fibaro

Returns

void

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

Parameters

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

Returns

void

Index