vxgproxyclient

0.1

Generated by Doxygen 1.8.17

Chapter 1

VXG Uplink Client Library

Chapter 2

Build System

2.0.1 Overview

VXG Uplink Client library uses `Meson` build system as a modern, fast and flexible build system that supports easy to set up and maintain a cross-compilation process.

It's recommended to refer to the `Meson` guide.

2.0.2 Build system installation

IMPORTANT: This projects requires Meson version >= 0.56.0

It's recommended to use `Ubuntu 20.04 LTS` distribution in development process but other distributions or operation systems are also supported by `Meson`.

Please refer to `Meson installation guide` to get and install `Meson`, preferable way to install `Meson` is `pip` method.

Quick install guide for Ubuntu 20.04. If you have an old version of meson already installed please remove it first.

```
sudo apt-get update
sudo apt-get install -y python3-pip git ninja-build curl tzdata python3-tz
pip3 install git+https://github.com/mesonbuild/meson@0.56.0
# pip3 puts meson main script into the $HOME/.local/bin/ directory, you need to
# add $HOME/.local/bin/ into your PATH environment variable, for bash shell you
# can run the following command and restart the shell session.
echo 'export PATH=$HOME/.local/bin:$PATH' » $HOME/.bashrc
# Check currently installed meson version
meson -v
```

Chapter 3

Application Development

3.1 Overview

An application that uses VXG Uplink Client Library should implement the Uplink::Proxy class derived from the base classes provided by the library:

- Uplink::Proxy - common implementation class, used for obtaining camera information such as serial number and MAC Address.

Any Proxy implementation should implement the get_serial_number, get_mac_address, and get_camera_info functions.

The library provides the stub implementation for most of the virtual methods of these classes, the stub implementation prints a log message about this method is not implemented and returns an error, the final application should implement all virtual methods on its own.

3.1.1 Linking application against the VXG Uplink Client Library

There are 3 possible ways of how to build and link your application

1. Building the application inside the VXG Uplink Client library's `Meson project`, the app will be assembled during the library project compilation in this case.
   You need to add a new executable target into the main `meson.build` file, please refer to the example app build target declaration:

   User must declare own executable target with a list of sources and dependencies, user may need to declare own dependencies if application requires it.

   This method is not recommended as it makes updating of the VXG Uplink Client library mostly not possible or very difficult for application developer

2. Building your app using your own build system and linking against the installed library.
   Running the `install` step from the compile section installs the binary libraries and headers into the directory you specified during the `setup` step, it also puts the `pkg-config`'s `.pc` files into the prefix directory which could be used by your own build system.

3. Preferred and recommended way of application development is to hold the app as a separate `Meson project` and use the VXG Uplink Client library as a `Meson subproject` of the application's `Meson project`.
   Using this approach gives the most flexible and convenient workflow for updating the VXG Uplink Library, all library dependencies will be promoted to the main project and will be also accessible by the application.
   How does it work

   - Assuming you have a `Meson` build system installed
   - Start a new `Meson project` with a following command:
     ```
     meson init -l cpp -n your-project-name
     ```
   - As a result of this command you should have the following files tree:
     ```
     |-- meson.build
     |-- your_project_name.cpp
     ```
   - Add VXG Uplink Client library as a `Meson subproject`
     All subprojects should be located in the `subprojects` directory so you have to create it first
     ```
     mkdir subprojects
     ```

   Now you have 2 options depending on how you want to store the VXG Uplink Client library sources:

   (a) If you want to store the VXG Uplink Client library as a files tree locally.

   - Create a symlink to the library path inside the subprojects dir:
     ```
     ln -s path/to/vxgproxyclient subprojects/vxgproxyclient
     ```

   Or you can just move vxgproxyclient directory inside the subprojects dir.

   - Create a library's `Meson` wrap file inside the subprojects dir, the name of the file should be the same as symlink you created in 1.1 and the content of the file should be:
     ```
     [wrap-file]
     directory = vxgproxyclient
     [provide]
     vxgproxyclient = vxgproxyclient_dep
     ```

   (b) If you want to store the library in a git repository you just need to create a wrap file with the content like below:
   ```
   [wrap-git]
   url=https://your-git-repo-url.com/path/vxgproxyclient.git
   # You can specify tag, branch or commit hash as revision
   revision=master
   [provide]
   vxgproxyclient = vxgproxyclient_dep
   ```

   You can find the example app `Meson project` in the example/app directory of the VXG Uplink library sources package.

Chapter 4

Library Compilation Guide

4.0.1 Library build process

Here is a compilation quickstart guide:

- First of all you need to have a build system and toolchain installed
- **Setup** the build directory
```
meson setup --prefix=path/to/install --strip -Dbuildtype=debug builddir/
# --prefix=path specifies the installation path
# --strip indicates that final binaries should be stripped
# -Dbuildtype= specifies the debug/release build type, please check the Meson docs about full list of
 the build types.
```
- **Build**
```
meson compile -C builddir
# Or
ninja -C builddir
```
- **Install**
```
meson install -C builddir
# Or
ninja -C builddir/ install
```
  As a result of the `install` step you should have the library compiled and installed into the prefix directory you specified during the `setup` step.
- **Clean**
```
ninja -C builddir clean
```
  Or you can just delete the builddir, you will need to `setup` it again in this case.
```
rm -rf builddir
```

4.0.2 Cross-compilation

- By default `Meson` builds project for the host platform, but it's also possible to cross-compile the library and your application using `Meson`.
- Full `Meson` cross-compilation documentation can be found `here`.
- The difference between the host compilation described above and the cross-compilation is the additional `--cross-file=path/to/cross-file.txt` flag for the Meson `Setup` step, the `Setup` command should look like below:
```
meson setup --prefix=path/to/install --strip -Dbuildtype=debug --cross-file=path/to/cross-file.txt
 builddir/
```
  `cross-file.txt` is the target platform description which in terms of Meson called a `cross-file`.
- `cross-file` example below is for the Debian provided `arm-linux-gnueabihf` toolchain installable using the Ubuntu's package manager command
```
sudo apt install g++-arm-linux-gnueabihf
```
- Example of the ARMv7 `cross-file` can be found in `/cross` directory:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Namespace Documentation

8.1 Uplink Namespace Reference

Data Structures

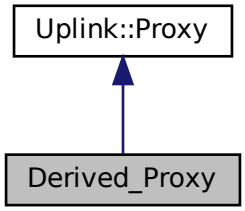
- class Proxy

Chapter 9

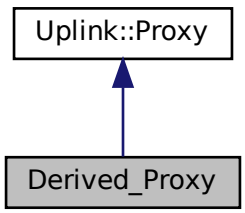
Data Structure Documentation

9.1 Derived_Proxy Class Reference

Inheritance diagram for Derived_Proxy:



Collaboration diagram for Derived_Proxy:



Public Member Functions

- int get_serial_number (char *ser_number) override

  [Get serial number implementation]
- int get_mac_address (char *mac_address) override

  [Get mac address implementation]
- int get_camera_info () override

  Get camera info function, responsible for retrieving camera S/N and MAC address.

9.1.1 Detailed Description

Definition at line 49 of file vxg_proxy_client.cc.

9.1.2 Member Function Documentation

9.1.2.1 get_camera_info()

```
int Derived_Proxy::get_camera_info ( )  [inline], [override], [virtual]
```

Get camera info function, responsible for retrieving camera S/N and MAC address.

Returns

0 if successful

Reimplemented from Uplink::Proxy.

Definition at line 70 of file vxg_proxy_client.cc.

9.1.2.2 get_mac_address()

```
int Derived_Proxy::get_mac_address (
            char * mac_address )  [inline], [override], [virtual]
```

[Get mac address implementation]

Reimplemented from Uplink::Proxy.

Definition at line 59 of file vxg_proxy_client.cc.

9.1.2.3 get_serial_number()

```
int Derived_Proxy::get_serial_number (
            char * ser_number )  [inline], [override], [virtual]
```

[Get serial number implementation]

Reimplemented from Uplink::Proxy.

Definition at line 52 of file vxg_proxy_client.cc.

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

- vxg_proxy_client.cc

9.2 forward_item Struct Reference

```
#include <agent/Proxy.h>
```

Data Fields

- char name [MAX_FORWARD_ITEM_NAME_LEN+1]
- char host [MAX_FORWARD_ITEM_HOST_LEN+1]
- Proto proto
- uint16_t port

9.2.1 Detailed Description

Definition at line 32 of file Proxy.h.

9.2.2 Field Documentation

9.2.2.1 host

```
char forward_item::host[MAX_FORWARD_ITEM_HOST_LEN+1]
```

Definition at line 34 of file Proxy.h.

9.2.2.2 name

```
char forward_item::name[MAX_FORWARD_ITEM_NAME_LEN+1]
```

Definition at line 33 of file Proxy.h.

9.2.2.3 port

```
uint16_t forward_item::port
```

Definition at line 36 of file Proxy.h.

9.2.2.4 proto

```
Proto forward_item::proto
```

Definition at line 35 of file Proxy.h.

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

- Proxy.h

9.3 msg Struct Reference

```
#include <agent/Proxy.h>
```

Data Fields

- void * payload
- size_t len

9.3.1 Detailed Description

Definition at line 39 of file Proxy.h.

9.3.2 Field Documentation

9.3.2.1 len

```
size_t msg::len
```

Definition at line 41 of file Proxy.h.

9.3.2.2 payload

```
void* msg::payload
```

Definition at line 40 of file Proxy.h.

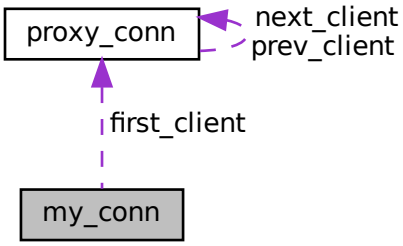
The documentation for this struct was generated from the following file:

- Proxy.h

9.4 my_conn Struct Reference

```
#include <agent/Proxy.h>
```

Collaboration diagram for my_conn:



Data Fields

- lws_sorted_usec_list_t sul
- struct lws * wsi
- uint16_t retry_count
- struct lws_ring * ring
- uint32_t tail
- char flow_controlled
- uint8_t write_consume_pending:1
- struct proxy_conn * first_client
- uint32_t total_msgs_in_client_rings
- void * obj

9.4.1 Detailed Description

Definition at line 60 of file Proxy.h.

9.4.2 Field Documentation

9.4.2.1 first_client

```
struct proxy_conn* my_conn::first_client
```

Definition at line 68 of file Proxy.h.

9.4.2.2 flow_controlled

```
char my_conn::flow_controlled
```

Definition at line 66 of file Proxy.h.

9.4.2.3 obj

```
void* my_conn::obj
```

Definition at line 70 of file Proxy.h.

9.4.2.4 retry_count

```
uint16_t my_conn::retry_count
```

Definition at line 63 of file Proxy.h.

9.4.2.5 ring

```
struct lws_ring* my_conn::ring
```

Definition at line 64 of file Proxy.h.

9.4.2.6 sul

```
lws_sorted_usec_list_t my_conn::sul
```

Definition at line 61 of file Proxy.h.

9.4.2.7 tail

```
uint32_t my_conn::tail
```

Definition at line 65 of file Proxy.h.

9.4.2.8 total_msgs_in_client_rings

```
uint32_t my_conn::total_msgs_in_client_rings
```

Definition at line 69 of file Proxy.h.

9.4.2.9 write_consume_pending

```
uint8_t my_conn::write_consume_pending
```

Definition at line 67 of file Proxy.h.

9.4.2.10 wsi

```
struct lws* my_conn::wsi
```

Definition at line 62 of file Proxy.h.

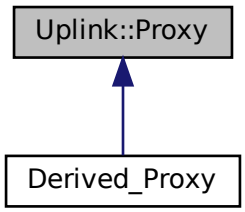
The documentation for this struct was generated from the following file:

- Proxy.h

9.5 Uplink::Proxy Class Reference

`#include <agent/Proxy.h>`

Inheritance diagram for Uplink::Proxy:



Public Member Functions

- Proxy ()
- virtual ~Proxy ()
- virtual int get_serial_number (char *ser_number)
- virtual int get_mac_address (char *mac_address)
- virtual int get_camera_info ()
- int start ()

  Start internal workflow, this is the main function which starts all internal connections.
- void stop ()

  Stop internal workflow, this is the main function which stops lws connection.
- void set_parameters (char *api_host, char *api_path, char *api_password, char *ws_host, char *ws_path, char *device_ser, char *token, int conn_port, int ssl_conn, **std::vector**< forward_item > *fwd_items)
- volatile int get_force_exit ()
- volatile int get_restart ()

9.5.1 Detailed Description

Definition at line 75 of file Proxy.h.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 Proxy()

`Uplink::Proxy::Proxy ( )`

Definition at line 3 of file Proxy.cpp.

9.5.2.2 ~Proxy()

```
virtual Uplink::Proxy::~Proxy ( )  [inline], [virtual]
```

Definition at line 126 of file Proxy.h.

9.5.3 Member Function Documentation

9.5.3.1 get_camera_info()

```
int Uplink::Proxy::get_camera_info ( )  [virtual]
```

Reimplemented in Derived_Proxy.

Definition at line 1054 of file Proxy.cpp.

9.5.3.2 get_force_exit()

```
volatile int Uplink::Proxy::get_force_exit ( )
```

Definition at line 1168 of file Proxy.cpp.

9.5.3.3 get_mac_address()

```
int Uplink::Proxy::get_mac_address (
            char * mac_address )  [virtual]
```

Reimplemented in Derived_Proxy.

Definition at line 1048 of file Proxy.cpp.

9.5.3.4 get_restart()

```
volatile int Uplink::Proxy::get_restart ( )
```

Definition at line 1173 of file Proxy.cpp.

9.5.3.5 get_serial_number()

```
int Uplink::Proxy::get_serial_number (
            char * ser_number )  [virtual]
```

Reimplemented in Derived_Proxy.

Definition at line 1042 of file Proxy.cpp.

9.5.3.6 set_parameters()

```
void Uplink::Proxy::set_parameters (
            char * api_host,
            char * api_path,
            char * api_password,
            char * ws_host,
            char * ws_path,
            char * device_ser,
            char * token,
            int conn_port,
            int ssl_conn,
             std::vector< forward_item > * fwd_items )
```

Definition at line 1135 of file Proxy.cpp.

9.5.3.7 start()

```
int Uplink::Proxy::start ( )
```

Start internal workflow, this is the main function which starts all internal connections.

Definition at line 1060 of file Proxy.cpp.

9.5.3.8 stop()

```
void Uplink::Proxy::stop ( )
```

Stop internal workflow, this is the main function which stops lws connection.

Definition at line 1123 of file Proxy.cpp.

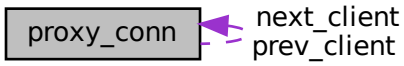
The documentation for this class was generated from the following files:

- Proxy.h
- Proxy.cpp

9.6 proxy_conn Struct Reference

`#include <agent/Proxy.h>`

Collaboration diagram for proxy_conn:



Data Fields

- struct lws * wsi_raw
- struct lws_ring * ring
- uint8_t forward_index
- client_id_t client_id
- uint32_t tail
- char flow_controlled
- char close_notification_sent
- uint8_t write_consume_pending:1
- struct proxy_conn * next_client
- struct proxy_conn * prev_client
- void * obj

9.6.1 Detailed Description

Definition at line 46 of file Proxy.h.

9.6.2 Field Documentation

9.6.2.1 client_id

`client_id_t proxy_conn::client_id`

Definition at line 50 of file Proxy.h.

9.6.2.2 close_notification_sent

```
char proxy_conn::close_notification_sent
```

Definition at line 53 of file Proxy.h.

9.6.2.3 flow_controlled

```
char proxy_conn::flow_controlled
```

Definition at line 52 of file Proxy.h.

9.6.2.4 forward_index

```
uint8_t proxy_conn::forward_index
```

Definition at line 49 of file Proxy.h.

9.6.2.5 next_client

```
struct proxy_conn* proxy_conn::next_client
```

Definition at line 55 of file Proxy.h.

9.6.2.6 obj

```
void* proxy_conn::obj
```

Definition at line 57 of file Proxy.h.

9.6.2.7 prev_client

```
struct proxy_conn* proxy_conn::prev_client
```

Definition at line 56 of file Proxy.h.

9.6.2.8 ring

```
struct lws_ring* proxy_conn::ring
```

Definition at line 48 of file Proxy.h.

9.6.2.9 tail

```
uint32_t proxy_conn::tail
```

Definition at line 51 of file Proxy.h.

9.6.2.10 write_consume_pending

```
uint8_t proxy_conn::write_consume_pending
```

Definition at line 54 of file Proxy.h.

9.6.2.11 wsi_raw

```
struct lws* proxy_conn::wsi_raw
```

Definition at line 47 of file Proxy.h.

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

- Proxy.h

Chapter 10

File Documentation

10.1 app-dev.md File Reference

10.2 build-system.md File Reference

10.3 compile.md File Reference

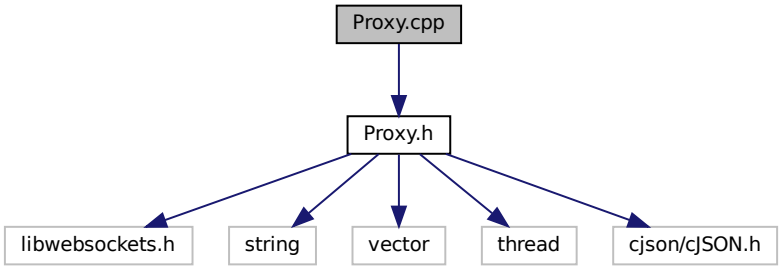
10.4 mainpage.md File Reference

10.5 meson.build File Reference

10.6 Proxy.cpp File Reference

```
#include "Proxy.h"
```

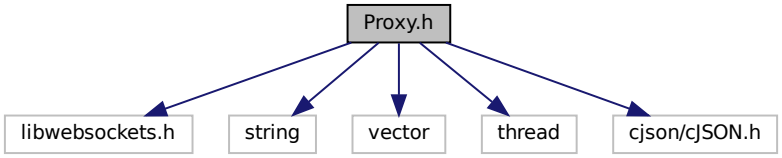
Include dependency graph for Proxy.cpp:



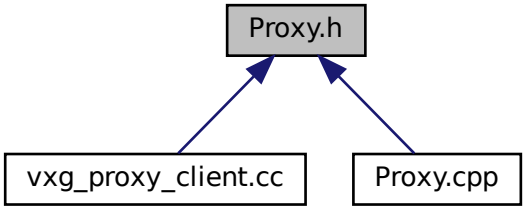
10.7 Proxy.h File Reference

```
#include <libwebsockets.h>
#include <string>
#include <vector>
#include <thread>
#include "cjson/cJSON.h"
```

Include dependency graph for Proxy.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct forward_item
- struct msg
- struct proxy_conn
- struct my_conn
- class Uplink::Proxy

Namespaces

- Uplink

Macros

- #define WEBSOCKET_BUFFER_SIZE
- #define AGENT_VERSION
- #define RING_DEPTH
- #define RING_DEPTH_CRITICAL
- #define RING_DEPTH_OK
- #define MAX_FORWARD_ITEM_NAME_LEN
- #define MAX_FORWARD_ITEM_HOST_LEN
- #define MAX_FORWARD_ITEMS
- #define MAX_CLIENT_CONNECTIONS

Typedefs

- typedef uint16_t client_id_t

Enumerations

- enum Proto { PROTO_NONE, PROTO_HTTP, PROTO_HTTPS, PROTO_TCP }

10.7.1 Macro Definition Documentation

10.7.1.1 AGENT_VERSION

```
#define AGENT_VERSION
```

Definition at line 14 of file Proxy.h.

10.7.1.2 MAX_CLIENT_CONNECTIONS

```
#define MAX_CLIENT_CONNECTIONS
```

Definition at line 23 of file Proxy.h.

10.7.1.3 MAX_FORWARD_ITEM_HOST_LEN

```
#define MAX_FORWARD_ITEM_HOST_LEN
```

Definition at line 21 of file Proxy.h.

10.7.1.4 MAX_FORWARD_ITEM_NAME_LEN

```
#define MAX_FORWARD_ITEM_NAME_LEN
```

Definition at line 20 of file Proxy.h.

10.7.1.5 MAX_FORWARD_ITEMS

```
#define MAX_FORWARD_ITEMS
```

Definition at line 22 of file Proxy.h.

10.7.1.6 RING_DEPTH

```
#define RING_DEPTH
```

Definition at line 17 of file Proxy.h.

10.7.1.7 RING_DEPTH_CRITICAL

```
#define RING_DEPTH_CRITICAL
```

Definition at line 18 of file Proxy.h.

10.7.1.8 RING_DEPTH_OK

```
#define RING_DEPTH_OK
```

Definition at line 19 of file Proxy.h.

10.7.1.9 WEBSOCKET_BUFFER_SIZE

```
#define WEBSOCKET_BUFFER_SIZE
```

Definition at line 10 of file Proxy.h.

10.7.2 Typedef Documentation

10.7.2.1 client_id_t

```
typedef uint16_t client_id_t
```

Definition at line 44 of file Proxy.h.

10.7.3 Enumeration Type Documentation

10.7.3.1 Proto

```
enum Proto
```

Enumerator

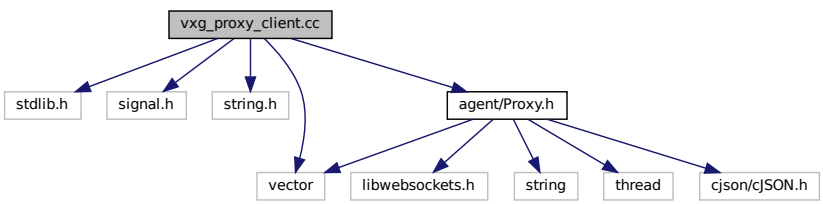
PROTO_NONE	
PROTO_HTTP	
PROTO_HTTPS	
PROTO_TCP	

Definition at line 25 of file Proxy.h.

10.8 vxg_proxy_client.cc File Reference

```
#include <stdlib.h>
#include <signal.h>
#include <string.h>
#include <vector>
#include <agent/Proxy.h>
```

Include dependency graph for vxg_proxy_client.cc:



Data Structures

- class Derived_Proxy

Functions

- static void signal_handler (int sig)
- bool parse_args (int argc, char **argv)

  Parse supplied command line arguments.
- int main (int argc, char **argv)

Variables

- volatile int reboot
- int n
- int i
- int forward_item_index
- char * p
- char * p_end
- char buf [MAX_FORWARD_ITEM_NAME_LEN+MAX_FORWARD_ITEM_HOST_LEN+21]
- **std::vector**< forward_item > app_forward_items (MAX_FORWARD_ITEMS+1)
- char api_host [128]
- char api_path [128]
- char api_password [128]
- char ws_host [256]
- char proxy_api_path [256]
- char ws_path [256]
- char device_sn [256]
- char token [1024 *8]
- int conn_port
- int ssl_conn
- static bool quit

10.8.1 Function Documentation

10.8.1.1 main()

```
int main (
            int argc,
            char ** argv )
```

Definition at line 254 of file vxg_proxy_client.cc.

10.8.1.2 parse_args()

```
bool parse_args (
            int argc,
            char ** argv )
```

Parse supplied command line arguments.

Returns

true if successful,
false if failure

Definition at line 97 of file vxg_proxy_client.cc.

10.8.1.3 signal_handler()

```
static void signal_handler (
            int sig )  [static]
```

Definition at line 36 of file vxg_proxy_client.cc.

10.8.2 Variable Documentation

10.8.2.1 api_host

```
char api_host[128]
```

Definition at line 14 of file vxg_proxy_client.cc.

10.8.2.2 api_password

```
char api_password[128]
```

Definition at line 16 of file vxg_proxy_client.cc.

10.8.2.3 api_path

```
char api_path[128]
```

Definition at line 15 of file vxg_proxy_client.cc.

10.8.2.4 app_forward_items

```
std::vector<forward_item> app_forward_items(MAX_FORWARD_ITEMS+1)
```

10.8.2.5 buf

```
char buf[MAX_FORWARD_ITEM_NAME_LEN+MAX_FORWARD_ITEM_HOST_LEN+21]
```

Definition at line 11 of file vxg_proxy_client.cc.

10.8.2.6 conn_port

```
int conn_port
```

Definition at line 30 of file vxg_proxy_client.cc.

10.8.2.7 device_sn

```
char device_sn[256]
```

Definition at line 20 of file vxg_proxy_client.cc.

10.8.2.8 forward_item_index

```
int forward_item_index
```

Definition at line 9 of file vxg_proxy_client.cc.

10.8.2.9 i

```
int i
```

Definition at line 8 of file vxg_proxy_client.cc.

10.8.2.10 n

```
int n
```

Definition at line 8 of file vxg_proxy_client.cc.

10.8.2.11 p

```
char* p
```

Definition at line 10 of file vxg_proxy_client.cc.

10.8.2.12 p_end

```
char * p_end
```

Definition at line 10 of file vxg_proxy_client.cc.

10.8.2.13 proxy_api_path

```
char proxy_api_path[256]
```

Definition at line 18 of file vxg_proxy_client.cc.

10.8.2.14 quit

```
bool quit  [static]
```

Definition at line 34 of file vxg_proxy_client.cc.

10.8.2.15 reboot

```
volatile int reboot
```

Definition at line 7 of file vxg_proxy_client.cc.

10.8.2.16 ssl_conn

```
int ssl_conn
```

Definition at line 31 of file vxg_proxy_client.cc.

10.8.2.17 token

```
char token[1024 *8]
```

Definition at line 21 of file vxg_proxy_client.cc.

10.8.2.18 ws_host

```
char ws_host[256]
```

Definition at line 17 of file vxg_proxy_client.cc.

10.8.2.19 ws_path

```
char ws_path[256]
```

Definition at line 19 of file vxg_proxy_client.cc.

Index