Snapdesk

Generated by Doxygen 1.9.4

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

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 compiler Namespace Reference

Classes

- class Compiler

  Tranform the source code into a executable tree.

5.2 database Namespace Reference

Classes

- class Csv

  This class represent and operates a csv file.
- class Database

  This class provide a simple API to manipulate storage files.

Functions

- std::string hash (std::string str)

5.2.1 Function Documentation

5.2.1.1 hash()

```
std::string database::hash (
            std::string str )
```

5.3 decoder Namespace Reference

Classes

- class Beacon_body

  Is the body of a beacon frame.
- class Big_number

  This class represent and manipulate indifined sized numbers using bytes.
- class Body

  This class set the base of all frame's body possible.
- class Frame

  The class that represent the whole frame.
- class Ie_node

  An element of a linked list of IEs.

5.4 executable_tree Namespace Reference

Classes

- class Cut_bit

  This Function cut the bits of the first arg from the second arg of the size of the third arg.
- class Cut_byte

  This Function cut the bytes of the first arg fril the secong arg of the size of the third arg.
- class Function

  Node that execute a function (that must be implemented as a child of Function class)
- class Getter

  Node that get dynamic information in the frame.
- class Node

  Represents a node of the executable tree.
- class Root

  The node that represent the root of the executable tree, it does not do anything but have one child.
- class Sha256

  This Function concatenates its arguments and.
- class Value

  Node that has a static value.

5.5 os_communicator Namespace Reference

Classes

- class Communicator

  The Communicator is the main class of the os_communicator component.

Chapter 6

Class Documentation

6.1 decoder::Beacon_body Class Reference

Is the body of a beacon frame.

`#include <management_body.hpp>`

Inheritance diagram for decoder::Beacon_body:

Collaboration diagram for decoder::Beacon_body:

Public Member Functions

- Beacon_body (uint8_t *raw_body_buffer, size_t raw_buffer_size)
  Construct a new Beacon_body object.
- ~Beacon_body ()
  Destroy the Beacon_body object.
- void print () const override
  Print the content of the beacon body.
- Big_number get_value (string field) const override
  Get the value of a specific field.

Additional Inherited Members

6.1.1 Detailed Description

Is the body of a beacon frame.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Beacon_body()

```
Beacon_body::Beacon_body (
            uint8_t * raw_body_buffer,
            size_t raw_buffer_size )
```

Construct a new Beacon_body object.

Parameters

raw_body_buffer	a buffer containing the exact beacon body
raw_buffer_size	the size of the buffer

6.1.2.2 ~Beacon_body()

```
Beacon_body::~Beacon_body ( )
```

Destroy the Beacon_body object.

6.1.3 Member Function Documentation

6.1.3.1 get_value()

```
Big_number Beacon_body::get_value (
            string field ) const  [override], [virtual]
```

Get the value of a specific field.

Parameters

field	the name of the field, or the element id corresponding to an IE

Returns

Big_number the value, or null if this do not exist

Implements decoder::Body.

6.1.3.2 print()

```
void Beacon_body::print ( ) const  [override], [virtual]
```

Print the content of the beacon body.

Implements decoder::Body.

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

- includes/decoder/management_body.hpp
- src/decoder/management_body.cpp

6.2 decoder::Big_number Class Reference

This class represent and manipulate indifined sized numbers using bytes.

`#include <big_number.hpp>`

Public Member Functions

- bool is_null () const

  Look if the number is null or not.
- void throw_if_null () const

  Throw an error if the number is null.
- std::string hex_string () const

  transform the number into a string containing the hex of it
- std::string char_string () const

  transform the number into a string containing the char of it
- size_t to_size_t () const

  Transform the number into a size_t number (only if the number can be contained in a size_t)
- void cut_byte (size_t from, size_t size)

  Cut the number to obtain only the given interval (unit byte)
- void cut_bit (size_t from, size_t size)

  Cut the number to obtain only the given interval (unit bit)
- Big_number copy () const

  Clone the Big_number object.
- Big_number & operator= (const Big_number _number)

  Copy the value of a Big_number object into another.

Static Public Member Functions

- static Big_number null ()

  Create a instance of Big_number containing the null value.
- static Big_number from_buffer (const uint8_t *buffer, const size_t buffer_size, const size_t number_size)

  Create a instance of Big_number containing the number from the given buffer.
- static Big_number from_buffer_inv (const uint8_t *buffer, const size_t buffer_size, const size_t number_size)

  Exactly the same as from_buffer() but with inverted bytes.
- static Big_number from_hex_string (const std::string string)

  Create a instance of Big_number containing the number from a given string.

6.2.1 Detailed Description

This class represent and manipulate indifined sized numbers using bytes.

6.2.2 Member Function Documentation

6.2.2.1 char_string()

```
std::string Big_number::char_string ( ) const
```

transform the number into a string containing the char of it

Returns

std::string the string containing the chars

6.2.2.2 copy()

```
Big_number Big_number::copy ( ) const
```

Clone the Big_number object.

Returns

Big_number the clone of the object

6.2.2.3 cut_bit()

```
void Big_number::cut_bit (
            size_t from,
            size_t size )
```

Cut the number to obtain only the given interval (unit bit)

Parameters

from	the first bit of the new number
size	the size of the new number

6.2.2.4 cut_byte()

```
void Big_number::cut_byte (
            size_t from,
            size_t size )
```

Cut the number to obtain only the given interval (unit byte)

Parameters

from	the first byte of the new number
size	the size of the new number

6.2.2.5 from_buffer()

```
Big_number Big_number::from_buffer (
            const uint8_t * buffer,
            const size_t buffer_size,
            const size_t number_size )  [static]
```

Create a instance of Big_number containing the number from the given buffer.

Parameters

buffer	the buffer containing the number, starting at the first number byte
buffer_size	the size of the buffer
number_size	the size of the number inside the buffer

Returns

Big_number the new instance with the value of the number in it

6.2.2.6 from_buffer_inv()

```
Big_number Big_number::from_buffer_inv (
            const uint8_t * buffer,
            const size_t buffer_size,
            const size_t number_size )  [static]
```

Exactly the same as from_buffer() but with inverted bytes.

Parameters

buffer	the buffer containing the number, starting at the first number byte
buffer_size	the size of the buffer
number_size	the size of the number inside the buffer

Returns

Big_number the new instance containing the value of the buffer

6.2.2.7 from_hex_string()

```
Big_number Big_number::from_hex_string (
            const std::string string )  [static]
```

Create a instance of Big_number containing the number from a given string.

Parameters

string	the string containing the bytes in hex form

Returns

Big_number the new instance containing the value of the number

6.2.2.8 hex_string()

```
std::string Big_number::hex_string ( ) const
```

transform the number into a string containing the hex of it

Returns

std::string the string containing the hexs

6.2.2.9 is_null()

```
bool Big_number::is_null ( ) const
```

Look if the number is null or not.

Returns

true if the number is null

false if the number is not null

6.2.2.10 null()

```
Big_number Big_number::null ( )  [static]
```

Create a instance of Big_number containing the null value.

Returns

Big_number the new instance

6.2.2.11 operator=()

```
Big_number & Big_number::operator= (
            const Big_number _number )
```

Copy the value of a Big_number object into another.

Parameters

_number	the Big_number object from where the value will be copied

Returns

Big_number& the Big_number where the value has been copied

6.2.2.12 throw_if_null()

```
void Big_number::throw_if_null ( ) const
```

Throw an error if the number is null.

6.2.2.13 to_size_t()

```
size_t Big_number::to_size_t ( ) const
```

Transform the number into a size_t number (only if the number can be contained in a size_t)

Returns

size_t the number corresponding to the Big_number

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

- includes/decoder/big_number.hpp
- src/decoder/big_number.cpp

6.3 decoder::Body Class Reference

This class set the base of all frame's body possible.

```
#include <frame.hpp>
```

Inheritance diagram for decoder::Body:

Public Member Functions

- Body (uint8_t *raw_body_buffer, size_t raw_buffer_size)
  Construct a new Body object.
- virtual void print () const =0
  Print the content of the body.
- virtual Big_number get_value (string field) const =0
  Get the value corresponding to the field.

Static Public Member Functions

- static Body * get_body (uint8_t *raw_body_buffer, size_t raw_buffer_size, size_t type, size_t sub_type)

  Give the Body objects corresponding to the type and subtype.

Protected Member Functions

- virtual void decode ()=0

  decode the body and fill the fields

Protected Attributes

- uint8_t * _raw_body_buffer

  The buffer that exactly contains the body.
- size_t _raw_buffer_size

  The size of the buffer.

6.3.1 Detailed Description

This class set the base of all frame's body possible.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Body()

```
Body::Body (
            uint8_t * raw_body_buffer,
            size_t raw_buffer_size )
```

Construct a new Body object.

Parameters

raw_body_buffer	a buffer containing the exact body
raw_buffer_size	the size of the buffer

6.3.3 Member Function Documentation

6.3.3.1 decode()

```
virtual void decoder::Body::decode ( )  [protected], [pure virtual]
```

decode the body and fill the fields

6.3.3.2 get_body()

```
Body * Body::get_body (
            uint8_t * raw_body_buffer,
            size_t raw_buffer_size,
            size_t type,
            size_t sub_type )  [static]
```

Give the Body objects corresponding to the type and subtype.

Parameters

raw_body_buffer	a buffer containing the exact body
raw_buffer_size	the size of the buffer
type	the type of the frame (management, control, or data)
sub_type	the subtype of the frame

Returns

Body* the new body corresponding to args

6.3.3.3 get_value()

```
virtual Big_number decoder::Body::get_value (
            string field ) const  [pure virtual]
```

Get the value corresponding to the field.

Parameters

field	the name of the field or the element id corresponding to the value

Returns

Big_number the value or null if not found

Implemented in decoder::Beacon_body.

6.3.3.4 print()

```
virtual void decoder::Body::print ( ) const  [pure virtual]
```

Print the content of the body.

Implemented in decoder::Beacon_body.

6.3.4 Member Data Documentation

6.3.4.1 _raw_body_buffer

```
uint8_t* decoder::Body::_raw_body_buffer  [protected]
```

The buffer that exactly contains the body.

6.3.4.2 _raw_buffer_size

```
size_t decoder::Body::_raw_buffer_size  [protected]
```

The size of the buffer.

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

- includes/decoder/frame.hpp
- src/decoder/frame.cpp

6.4 os_communicator::Communicator Class Reference

The Communicator is the main class of the os_communicator component.

```
#include <os_communicator.hpp>
```

Public Member Functions

- Communicator (string file_name)
  Construct a new Communicator object.
- bool exist ()
  Say if the given file exist.
- size_t in_buffer (uint8_t *buffer, const size_t buffer_size)
  Copy the content of the file into the given buffer.
- string get_line (size_t line_number) const
  Get the line number line_number from the file.
- void add_line (string line)
  Add a line at the end of the file.
- void replace_line (size_t line_number, string new_line)
  Replace the line line_number with the line new_line.
- void new_file ()
  Create an empty file.

Static Public Member Functions

- static string get_current_date ()

  Get the current date.
- static void notify (string message)

  Notify the user with a message.
- static void sleep (size_t seconds)

  Pause the process for an amount of time.
- static void create_folder (string folder_name)

  Create a folder.
- static size_t get_file_type (string name)

  Get the type of a file (file, folder, nothing)

6.4.1 Detailed Description

The Communicator is the main class of the os_communicator component.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Communicator()

```
os_communicator::Communicator::Communicator (
            string file_name )
```

Construct a new Communicator object.

Parameters

file_name	The name of the file to interact with

6.4.3 Member Function Documentation

6.4.3.1 add_line()

```
void os_communicator::Communicator::add_line (
            string line )
```

Add a line at the end of the file.

Parameters

line	the line to add to the end of the file

6.4.3.2 create_folder()

```
void os_communicator::Communicator::create_folder (
            string folder_name )  [static]
```

Create a folder.

Parameters

folder_name	the name of the folder

6.4.3.3 exist()

```
bool os_communicator::Communicator::exist ( )
```

Say if the given file exist.

Returns

true if the file exist

false if the file does not exist

6.4.3.4 get_current_date()

```
string os_communicator::Communicator::get_current_date ( )  [static]
```

Get the current date.

Returns

string the current date

6.4.3.5 get_file_type()

```
size_t os_communicator::Communicator::get_file_type (
            string name )  [static]
```

Get the type of a file (file, folder, nothing)

Parameters

name	the name of the file

Returns

size_t the type of the file

6.4.3.6 get_line()

```
string os_communicator::Communicator::get_line (
            size_t line_number ) const
```

Get the line number line_number from the file.

Parameters

line_number	the number of the line from file to get

Returns

string the line

6.4.3.7 in_buffer()

```
size_t os_communicator::Communicator::in_buffer (
            uint8_t * buffer,
            const size_t buffer_size )
```

Copy the content of the file into the given buffer.

Parameters

buffer	the buffer where the data will be copied
buffer_size	the size of the buffer

Returns

size_t the number of byte read

6.4.3.8 new_file()

```
void os_communicator::Communicator::new_file ( )
```

Create an empty file.

6.4.3.9 notify()

```
void os_communicator::Communicator::notify (
            string message )  [static]
```

Notify the user with a message.

Parameters

message	the message that will be displayed

6.4.3.10 replace_line()

```
void os_communicator::Communicator::replace_line (
            size_t line_number,
            string new_line )
```

Replace the line line_number with the line new_line.

Parameters

line_number	the number of the file to replace
new_line	the line that will replace the old one

6.4.3.11 sleep()

```
void os_communicator::Communicator::sleep (
            size_t seconds )  [static]
```

Pause the process for an amount of time.

Parameters

seconds	the number of seconds to pause the process

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

- includes/os_communicator/os_communicator.hpp
- src/os_communicator/os_communicator.cpp

6.5 compiler::Compiler Class Reference

Tranform the source code into a executable tree.

`#include <compiler.hpp>`

Public Member Functions

- Compiler (const os_communicator::Communicator *communicator)

  Construct a new Compiler object.
- executable_tree::Node * get_executable_tree () const

  Compile the source code to get the executable tree.

6.5.1 Detailed Description

Tranform the source code into a executable tree.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Compiler()

```
compiler::Compiler::Compiler (
            const os_communicator::Communicator * communicator )  [inline]
```

Construct a new Compiler object.

Parameters

communicator	Communicator to source code

6.5.3 Member Function Documentation

6.5.3.1 get_executable_tree()

```
executable_tree::Node * Compiler::get_executable_tree ( ) const
```

Compile the source code to get the executable tree.

Returns

executable_tree::Node* the executable tree corresponding to the source code

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

- includes/compiler/compiler.hpp
- src/compiler/compiler.cpp

6.6 database::Csv Class Reference

This class represent and operates a csv file.

`#include <core.hpp>`

Public Member Functions

- Csv (std::string file_name)
  Construct a new Csv object on a file that does exist.
- ~Csv ()
  Destroy the Csv object.
- void push_row (std::vector< std::string > values)
  Create a new entry in the file.
- void replace_row (size_t row_number, std::vector< std::string > values)
  Replace a row by another.
- std::string get_cell (size_t row_number, size_t column_number)
  Get the value of a cell.
- size_t get_row_number (std::string key_value)
  Get the row number of a given key value.
- size_t get_column_number (std::string column_name)
  Get the column number from its name.
- std::vector< std::string > get_all_keys ()
  Get all key values of a file.
- std::vector< std::string > get_all_keys_with_value (size_t column_number, std::string value)
  Get all key values that has a value on a given column.

Static Public Member Functions

- static Csv * create (std::string file_name, std::vector< std::string > column_names, size_t key_column_↩ number)
  Construct a new Csv object on a file that does not exist.

6.6.1 Detailed Description

This class represent and operates a csv file.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Csv()

```
database::Csv::Csv (
            std::string file_name )
```

Construct a new Csv object on a file that does exist.

Parameters

file_name	the name of the storage file (without extentions)

6.6.2.2 ~Csv()

```
database::Csv::~Csv ( )
```

Destroy the Csv object.

6.6.3 Member Function Documentation

6.6.3.1 create()

```
Csv * database::Csv::create (
            std::string file_name,
            std::vector< std::string > column_names,
            size_t key_column_number )  [static]
```

Construct a new Csv object on a file that does not exist.

Parameters

file_name	the filename of the new storage file (without extentions)
column_names	the names of the columns
key_column_number	the column number of the key

Returns

Csv* the new Csv instance of the new storage file

6.6.3.2 get_all_keys()

```
std::vector< std::string > database::Csv::get_all_keys ( )
```

Get all key values of a file.

Returns

std::vector<std::string> a vector containing all the key values

6.6.3.3 get_all_keys_with_value()

```
std::vector< std::string > database::Csv::get_all_keys_with_value (
            size_t column_number,
            std::string value )
```

Get all key values that has a value on a given column.

Parameters

column_number	the number of the column to look for matches
value	the value to compare

Returns

std::vector<std::string> a vector containing all the key values

6.6.3.4 get_cell()

```
std::string database::Csv::get_cell (
            size_t row_number,
            size_t column_number )
```

Get the value of a cell.

Parameters

row_number	the row number of the cell
column_number	the column number of the cell

Returns

std::string the value of the cell

6.6.3.5 get_column_number()

```
size_t database::Csv::get_column_number (
            std::string column_name )
```

Get the column number from its name.

Parameters

column_name	the name of the column

Returns

size_t the column number, or -1 if not found

6.6.3.6 get_row_number()

```
size_t database::Csv::get_row_number (
            std::string key_value )
```

Get the row number of a given key value.

Parameters

key_value	the value to match with the key

Returns

size_t the row number of the given key value, or -1 if not found

6.6.3.7 push_row()

```
void database::Csv::push_row (
            std::vector< std::string > values )
```

Create a new entry in the file.

Parameters

values	the vector of values to add

6.6.3.8 replace_row()

```
void database::Csv::replace_row (
            size_t row_number,
            std::vector< std::string > values )
```

Replace a row by another.

Parameters

row_number	the number of the row to replace
values	the vector of values that will replace

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

- includes/database/core.hpp
- src/database/csv.cpp

6.7 executable_tree::Cut_bit Class Reference

This Function cut the bits of the first arg from the second arg of the size of the third arg.

`#include <function_node.hpp>`

Inheritance diagram for executable_tree::Cut_bit:

Collaboration diagram for executable_tree::Cut_bit:

Public Member Functions

- std::string to_string (size_t depth) const override

  get the string representing the node
- std::string get_value (const decoder::Frame *target_frame) const override

  Get the value of the node.

Additional Inherited Members

6.7.1 Detailed Description

This Function cut the bits of the first arg from the second arg of the size of the third arg.

6.7.2 Member Function Documentation

6.7.2.1 get_value()

```
std::string Cut_bit::get_value (
            const decoder::Frame * target_frame ) const  [override], [virtual]
```

Get the value of the node.

Parameters

target_frame	the frame that is analysed by the executable tree

Returns

std::string the value returned by the node

Reimplemented from executable_tree::Node.

6.7.2.2 to_string()

```
std::string Cut_bit::to_string (
            size_t depth ) const  [override], [virtual]
```

get the string representing the node

Parameters

depth	the depth of the node in the executable tree

Returns

std::string the string representing the node

Reimplemented from executable_tree::Node.

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

- includes/compiler/function_node.hpp
- src/compiler/function_node.cpp

6.8 executable_tree::Cut_byte Class Reference

This Function cut the bytes of the first arg fril the secong arg of the size of the third arg.

```
#include <function_node.hpp>
```

Inheritance diagram for executable_tree::Cut_byte:

Collaboration diagram for executable_tree::Cut_byte:

Public Member Functions

- std::string to_string (size_t depth) const override
  get the string representing the node
- std::string get_value (const decoder::Frame *target_frame) const override
  Get the value of the node.

Additional Inherited Members

6.8.1 Detailed Description

This Function cut the bytes of the first arg fril the secong arg of the size of the third arg.

6.8.2 Member Function Documentation

6.8.2.1 get_value()

```
std::string Cut_byte::get_value (
            const decoder::Frame * target_frame ) const  [override], [virtual]
```

Get the value of the node.

Parameters

target_frame	the frame that is analysed by the executable tree

Returns

std::string the value returned by the node

Reimplemented from executable_tree::Node.

6.8.2.2 to_string()

```
std::string Cut_byte::to_string (
            size_t depth ) const  [override], [virtual]
```

get the string representing the node

Parameters

depth	the depth of the node in the executable tree

Returns

std::string the string representing the node

Reimplemented from executable_tree::Node.

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

- includes/compiler/function_node.hpp
- src/compiler/function_node.cpp

6.9 database::Database Class Reference

This class provide a simple API to manipulate storage files.

`#include <core.hpp>`

Public Member Functions

- Database (std::string ssid, std::string code_file_name)

  Construct a new Database object.
- ~Database ()

  Destroy the Database object.
- std::string get_cell (std::string key_value, std::string column_name)

  Get a cell given the value of its key and its column number.
- std::vector< std::string > get_key_of_entries (std::string column_name, std::string cell_value)

  Get all keys values corresponding to a given cell value.
- void add_entry (std::vector< std::string > values)

  Add an entry to the storage file.
- void replace_entry (std::string key_value, std::vector< std::string > values)

  Replace an entry by another.

6.9.1 Detailed Description

This class provide a simple API to manipulate storage files.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Database()

```
database::Database::Database (
            std::string ssid,
            std::string code_file_name )
```

Construct a new Database object.

Parameters

ssid	The SSID of the analysed frames
code_file_name	The file name of the used code

6.9.2.2 ~Database()

```
database::Database::~Database ( )
```

Destroy the Database object.

6.9.3 Member Function Documentation

6.9.3.1 add_entry()

```
void database::Database::add_entry (
            std::vector< std::string > values )
```

Add an entry to the storage file.

Parameters

values	the vector of values corresponding to the new entry

6.9.3.2 get_cell()

```
std::string database::Database::get_cell (
            std::string key_value,
            std::string column_name )
```

Get a cell given the value of its key and its column number.

Parameters

key_value	the value of the key
column_name	the column name of the cell

Returns

std::string the value of the cell

6.9.3.3 get_key_of_entries()

```
std::vector< std::string > database::Database::get_key_of_entries (
            std::string column_name,
            std::string cell_value )
```

Get all keys values corresponding to a given cell value.

Parameters

column_name	the name of the column to compare
cell_value	the value to compare

Returns

std::vector<std::string> the vector of key values

6.9.3.4 replace_entry()

```
void database::Database::replace_entry (
            std::string key_value,
            std::vector< std::string > values )
```

Replace an entry by another.

Parameters

key_value	the key value of the entry to replace
values	the vector of new values to replace the ols entry with

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

- includes/database/core.hpp
- src/database/database.cpp

6.10 decoder::Frame Class Reference

The class that represent the whole frame.

```
#include <frame.hpp>
```

Collaboration diagram for decoder::Frame:

Public Member Functions

- Frame (os_communicator::Communicator *_communicator)

  Construct a new Frame object.
- ~Frame ()

  Destroy the Frame object.
- bool get_is_decoded () const

  Get the value of is_decoded.
- void update_raw_data ()

  Get a new frame from the character device and put it in raw_frame_buffer.
- void print_raw_data ()

  Print the raw_frame_buffer as hex bytes.
- virtual void decode ()

  Decode the raw_frame_buffer to fill the different fields.
- virtual void print () const

  Print the content of the decoded frame.
- Big_number get_value (string field) const

  Get the value corresponding to the field.

Protected Attributes

- uint8_t * raw_frame_buffer

  the buffer containing the frame that is not decoded
- size_t raw_buffer_size

  the size of the buffer
- size_t raw_frame_size

  the size of the frame in the buffer
- bool is_decoded

  true if the buffer is decoded, false if not
- bool has_raw_data

  true if raw_frame_buffer is filled
- Big_number frame_control = Big_number::null()
- Big_number duration = Big_number::null()
- Big_number destination_address = Big_number::null()
- Big_number source_address = Big_number::null()
- Big_number bssid = Big_number::null()
- Big_number sequence_control = Big_number::null()
- Big_number frame_check_sum = Big_number::null()
- Body * body = nullptr

  the body of the frame

6.10.1 Detailed Description

The class that represent the whole frame.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Frame()

```
Frame::Frame (
            os_communicator::Communicator * _communicator )
```

Construct a new Frame object.

Parameters

_communicator	Communicator on the character device

6.10.2.2 ~Frame()

```
Frame::~Frame ( )
```

Destroy the Frame object.

6.10.3 Member Function Documentation

6.10.3.1 decode()

```
void Frame::decode ( )  [virtual]
```

Decode the raw_frame_buffer to fill the different fields.

6.10.3.2 get_is_decoded()

```
bool Frame::get_is_decoded ( ) const
```

Get the value of is_decoded.

Returns

true if is_decoded is true

false if is_decoded is false

6.10.3.3 get_value()

```
Big_number Frame::get_value (
            string field ) const
```

Get the value corresponding to the field.

Parameters

field	the name of the field that we want to get

Returns

Big_number the field value, or null if not found

6.10.3.4 print()

```
void Frame::print ( ) const  [virtual]
```

Print the content of the decoded frame.

6.10.3.5 print_raw_data()

```
void Frame::print_raw_data ( )
```

Print the raw_frame_buffer as hex bytes.

6.10.3.6 update_raw_data()

```
void Frame::update_raw_data ( )
```

Get a new frame from the character device and put it in raw_frame_buffer.

6.10.4 Member Data Documentation

6.10.4.1 body

```
Body* decoder::Frame::body = nullptr  [protected]
```

the body of the frame

6.10.4.2 bssid

```
Big_number decoder::Frame::bssid = Big_number::null()  [protected]
```

6.10.4.3 destination_address

```
Big_number decoder::Frame::destination_address = Big_number::null()  [protected]
```

6.10.4.4 duration

```
Big_number decoder::Frame::duration = Big_number::null()  [protected]
```

6.10.4.5 frame_check_sum

```
Big_number decoder::Frame::frame_check_sum = Big_number::null()  [protected]
```

6.10.4.6 frame_control

```
Big_number decoder::Frame::frame_control = Big_number::null()  [protected]
```

6.10.4.7 has_raw_data

```
bool decoder::Frame::has_raw_data  [protected]
```

true if raw_frame_buffer is filled

6.10.4.8 is_decoded

```
bool decoder::Frame::is_decoded  [protected]
```

true if the buffer is decoded, false if not

6.10.4.9 raw_buffer_size

```
size_t decoder::Frame::raw_buffer_size  [protected]
```

the size of the buffer

6.10.4.10 raw_frame_buffer

```
uint8_t* decoder::Frame::raw_frame_buffer  [protected]
```

the buffer containing the frame that is not decoded

6.10.4.11 raw_frame_size

```
size_t decoder::Frame::raw_frame_size  [protected]
```

the size of the frame in the buffer

6.10.4.12 sequence_control

```
Big_number decoder::Frame::sequence_control = Big_number::null()  [protected]
```

6.10.4.13 source_address

```
Big_number decoder::Frame::source_address = Big_number::null()  [protected]
```

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

- includes/decoder/frame.hpp
- src/decoder/frame.cpp

6.11 executable_tree::Function Class Reference

Node that execute a function (that must be implemented as a child of Function class)

```
#include <node.hpp>
```

Inheritance diagram for executable_tree::Function:

Collaboration diagram for executable_tree::Function:

Public Member Functions

- ~Function ()

  Destroy the Function object.
- void add_node (Node *arg) override

  Add a child node to the current node.
- std::string to_string (size_t depth) const override

  get the string representing the node

Protected Attributes

- std::vector< Node * > args

  The vector of arguments of the function.

6.11.1 Detailed Description

Node that execute a function (that must be implemented as a child of Function class)

6.11.2 Constructor & Destructor Documentation

6.11.2.1 ~Function()

```
executable_tree::Function::~Function ( )  [inline]
```

Destroy the Function object.

6.11.3 Member Function Documentation

6.11.3.1 add_node()

```
void Function::add_node (
            Node * arg )  [override], [virtual]
```

Add a child node to the current node.

Parameters

arg	The node to add

Reimplemented from executable_tree::Node.

6.11.3.2 to_string()

```
std::string Function::to_string (
            size_t depth ) const  [override], [virtual]
```

get the string representing the node

Parameters

depth	the depth of the node in the executable tree

Returns

std::string the string representing the node

Reimplemented from executable_tree::Node.

6.11.4 Member Data Documentation

6.11.4.1 args

```
std::vector<Node*> executable_tree::Function::args  [protected]
```

The vector of arguments of the function.

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

- includes/compiler/node.hpp
- src/compiler/node.cpp

6.12 executable_tree::Getter Class Reference

Node that get dynamic information in the frame.

```
#include <node.hpp>
```

Inheritance diagram for executable_tree::Getter:

Collaboration diagram for executable_tree::Getter:

Public Member Functions

- Getter (const std::string field_name)

  Construct a new Getter object.
- std::string get_value (const decoder::Frame *target_frame) const override

  Get the value of the node.
- void add_node (Node *arg) override

  Add a child node to the current node.
- std::string to_string (size_t depth) const override

  get the string representing the node

6.12.1 Detailed Description

Node that get dynamic information in the frame.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 Getter()

```
executable_tree::Getter::Getter (
            const std::string field_name )  [inline]
```

Construct a new Getter object.

Parameters

field_name	the name of the field where to get information

6.12.3 Member Function Documentation

6.12.3.1 add_node()

```
void Getter::add_node (
            Node * arg )  [override], [virtual]
```

Add a child node to the current node.

Parameters

arg	The node to add

Reimplemented from executable_tree::Node.

6.12.3.2 get_value()

```
std::string Getter::get_value (
            const decoder::Frame * target_frame ) const  [override], [virtual]
```

Get the value of the node.

Parameters

target_frame	the frame that is analysed by the executable tree

Returns

std::string the value returned by the node

Reimplemented from executable_tree::Node.

6.12.3.3 to_string()

```
std::string Getter::to_string (
            size_t depth ) const  [override], [virtual]
```

get the string representing the node

Parameters

depth	the depth of the node in the executable tree

Returns

std::string the string representing the node

Reimplemented from executable_tree::Node.

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

- includes/compiler/node.hpp
- src/compiler/node.cpp

6.13 decoder::le_node Class Reference

An element of a linked list of IEs.

```
#include <ie.hpp>
```

Public Member Functions

- Ie_node (uint8_t element_id, Big_number value)

  Construct a new Ie_node object.
- ~Ie_node ()

  Destroy the Ie_node object.
- void add (Ie_node *new_element)

  Add a node to the linked list.
- Big_number get_value (std::string field) const

  Get the value of corresponding to a field.
- void print ()

  Print the IE.

6.13.1 Detailed Description

An element of a linked list of IEs.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Ie_node()

```
Ie_node::Ie_node (
            uint8_t element_id,
            Big_number value )
```

Construct a new Ie_node object.

Parameters

element_id	the element id of the new IE
value	the value of the new IE

6.13.2.2 ~Ie_node()

```
Ie_node::~Ie_node ( )
```

Destroy the Ie_node object.

6.13.3 Member Function Documentation

6.13.3.1 add()

```
void Ie_node::add (
            Ie_node * new_element )
```

Add a node to the linked list.

Parameters

new_element	the new node to add

6.13.3.2 get_value()

```
Big_number Ie_node::get_value (
            std::string field ) const
```

Get the value of corresponding to a field.

Parameters

field	the name of the field or its element id

Returns

Big_number the value that correcpond to the field or null if none

6.13.3.3 print()

```
void Ie_node::print ( )
```

Print the IE.

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

- includes/decoder/ie.hpp
- src/decoder/ie.cpp

6.14 executable_tree::Node Class Reference

Represents a node of the executable tree.

```
#include <node.hpp>
```

Inheritance diagram for executable_tree::Node:

Public Member Functions

- virtual std::string get_value (const decoder::Frame *target_frame) const

  Get the value of the node.
- virtual void add_node (Node *arg)

  Add a child node to the current node.
- virtual std::string to_string (size_t depth) const

  get the string representing the node

6.14.1 Detailed Description

Represents a node of the executable tree.

6.14.2 Member Function Documentation

6.14.2.1 add_node()

```
void Node::add_node (
            Node * arg )  [virtual]
```

Add a child node to the current node.

Parameters

arg	The node to add

Reimplemented in executable_tree::Root, executable_tree::Function, executable_tree::Value, and executable_tree::Getter.

6.14.2.2 get_value()

```
std::string Node::get_value (
            const decoder::Frame * target_frame ) const  [virtual]
```

Get the value of the node.

Parameters

target_frame	the frame that is analysed by the executable tree

Returns

std::string the value returned by the node

Reimplemented in executable_tree::Sha256, executable_tree::Cut_bit, executable_tree::Cut_byte, executable_tree::Root, executable_tree::Value, and executable_tree::Getter.

6.14.2.3 to_string()

```
std::string Node::to_string (
            size_t depth ) const  [virtual]
```

get the string representing the node

Parameters

depth	the depth of the node in the executable tree

Returns

std::string the string representing the node

Reimplemented in executable_tree::Sha256, executable_tree::Cut_bit, executable_tree::Cut_byte, executable_tree::Root, executable_tree::Function, executable_tree::Value, and executable_tree::Getter.

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

- includes/compiler/node.hpp
- src/compiler/node.cpp

6.15 executable_tree::Root Class Reference

The node that represent the root of the executable tree, it does not do anything but have one child.

```
#include <node.hpp>
```

Inheritance diagram for executable_tree::Root:

Collaboration diagram for executable_tree::Root:

Public Member Functions

- ~Root ()
  Destroy the Root object.
- std::string get_value (const decoder::Frame *target_frame) const override
  Get the value of the node.
- void add_node (Node *arg) override
  Add a child node to the current node.
- std::string to_string (size_t depth) const override
  get the string representing the node

6.15.1 Detailed Description

The node that represent the root of the executable tree, it does not do anything but have one child.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 ~Root()

```
executable_tree::Root::~Root ( )  [inline]
```

Destroy the Root object.

6.15.3 Member Function Documentation

6.15.3.1 add_node()

```
void Root::add_node (
            Node * arg )  [override], [virtual]
```

Add a child node to the current node.

Parameters

arg	The node to add

Reimplemented from executable_tree::Node.

6.15.3.2 get_value()

```
std::string Root::get_value (
            const decoder::Frame * target_frame ) const  [override], [virtual]
```

Get the value of the node.

Parameters

target_frame	the frame that is analysed by the executable tree

Returns

std::string the value returned by the node

Reimplemented from executable_tree::Node.

6.15.3.3 to_string()

```
std::string Root::to_string (
            size_t depth ) const  [override], [virtual]
```

get the string representing the node

Parameters

depth	the depth of the node in the executable tree

Returns

std::string the string representing the node

Reimplemented from executable_tree::Node.

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

- includes/compiler/node.hpp
- src/compiler/node.cpp

6.16 executable_tree::Sha256 Class Reference

This Function concatenates its arguments and.

```
#include <function_node.hpp>
```

Inheritance diagram for executable_tree::Sha256:

Collaboration diagram for executable_tree::Sha256:

Public Member Functions

- std::string to_string (size_t depth) const override
  get the string representing the node
- std::string get_value (const decoder::Frame *target_frame) const override
  Get the value of the node.

Additional Inherited Members

6.16.1 Detailed Description

This Function concatenates its arguments and.

6.16.2 Member Function Documentation

6.16.2.1 get_value()

```
std::string Sha256::get_value (
            const decoder::Frame * target_frame ) const  [override], [virtual]
```

Get the value of the node.

Parameters

target_frame	the frame that is analysed by the executable tree

Returns

std::string the value returned by the node

Reimplemented from executable_tree::Node.

6.16.2.2 to_string()

```
std::string Sha256::to_string (
            size_t depth ) const  [override], [virtual]
```

get the string representing the node

Parameters

depth	the depth of the node in the executable tree

Returns

std::string the string representing the node

Reimplemented from executable_tree::Node.

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

- includes/compiler/function_node.hpp
- src/compiler/function_node.cpp

6.17 executable_tree::Value Class Reference

Node that has a static value.

`#include <node.hpp>`

Inheritance diagram for executable_tree::Value:

Collaboration diagram for executable_tree::Value:

Public Member Functions

- Value (std::string value)

  Construct a new Value object.
- std::string get_value (const decoder::Frame *target_frame) const override

  Get the value of the node.
- void add_node (Node *arg) override

  Add a child node to the current node.
- std::string to_string (size_t depth) const override

  get the string representing the node

6.17.1 Detailed Description

Node that has a static value.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 Value()

```
executable_tree::Value::Value (
            std::string value )  [inline]
```

Construct a new Value object.

Parameters

value	the fixed value of the Node

6.17.3 Member Function Documentation

6.17.3.1 add_node()

```
void Value::add_node (
            Node * arg )  [override], [virtual]
```

Add a child node to the current node.

Parameters

arg	The node to add

Reimplemented from executable_tree::Node.

6.17.3.2 get_value()

```
std::string Value::get_value (
            const decoder::Frame * target_frame ) const  [override], [virtual]
```

Get the value of the node.

Parameters

target_frame	the frame that is analysed by the executable tree

Returns

std::string the value returned by the node

Reimplemented from executable_tree::Node.

6.17.3.3 to_string()

```
std::string Value::to_string (
            size_t depth ) const  [override], [virtual]
```

get the string representing the node

Parameters

depth	the depth of the node in the executable tree

Returns

std::string the string representing the node

Reimplemented from executable_tree::Node.

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

- includes/compiler/node.hpp
- src/compiler/node.cpp

Chapter 7

File Documentation

7.1 includes/compiler/compiler.hpp File Reference

```
#include "os_communicator/os_communicator.hpp"
#include "compiler/node.hpp"
#include "compiler/function_node.hpp"
#include "decoder/frame.hpp"
```

Include dependency graph for compiler.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class compiler::Compiler

  Tranform the source code into a executable tree.

Namespaces

- namespace compiler

7.2 compiler.hpp

Go to the documentation of this file.

```
1 #ifndef COMPILER_HPP
2 #define COMPILER_HPP
3
4 #include "os_communicator/os_communicator.hpp"
5 #include "compiler/node.hpp"
6 #include "compiler/function_node.hpp"
7 #include "decoder/frame.hpp"
8
9 namespace compiler {
15     class Compiler {
16         private:
17             const os_communicator::Communicator *_communicator;
18
19             // return the next line to read and fill the current_node
27             size_t read_line(executable_tree::Node *current_node, size_t next_line) const;
28
29         public:
35             Compiler(const os_communicator::Communicator *communicator)
36             :  _communicator(communicator) {
37                 if(!_communicator)
38                     throw invalid_argument("No communicator given");
39             };
40
46             executable_tree::Node *get_executable_tree() const;
47     };
48 }
49
50 #endif
```

7.3 includes/compiler/function_node.hpp File Reference

```
#include "compiler/node.hpp"
#include "decoder/big_number.hpp"
#include <openssl/evp.h>
#include <string>
```

Include dependency graph for function_node.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class executable_tree::Sha256

  This Function concatenates its arguments and.
- class executable_tree::Cut_bit

  This Function cut the bits of the first arg from the second arg of the size of the third arg.
- class executable_tree::Cut_byte

  This Function cut the bytes of the first arg fril the secong arg of the size of the third arg.

Namespaces

- namespace executable_tree

7.3.1 Detailed Description

Author

Pagano Florian

Version

0.1

Date

2025

Copyright

Copyright (c) 2025

7.4 function_node.hpp

Go to the documentation of this file.

```
1
11 #ifndef FUNCTION_NODE_HPP
12 #define FUNCTION_NODE_HPP
13
14 #include "compiler/node.hpp"
15 #include "decoder/big_number.hpp"
16
17 #include <openssl/evp.h>
18 #include <string>
19
20 namespace executable_tree{
26     class Sha256 :  public Function {
27         public:
28             std::string to_string(size_t depth) const override;
29
30             std::string get_value(const decoder::Frame *target_frame) const override;
31     } ;
32
37     class Cut_bit :  public Function {
38         public:
39             std::string to_string(size_t depth) const override;
40
41             std::string get_value(const decoder::Frame *target_frame) const override;
42     } ;
43
48     class Cut_byte :  public Function {
49         public:
50             std::string to_string(size_t depth) const override;
51
52             std::string get_value(const decoder::Frame *target_frame) const override;
53     } ;
54 }
55
56 #endif
```

7.5 includes/compiler/node.hpp File Reference

This file contains base class of a executable tree.

```
#include "decoder/frame.hpp"
#include <vector>
#include <string>
```

Include dependency graph for node.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class executable_tree::Node

  Represents a node of the executable tree.
- class executable_tree::Root

  The node that represent the root of the executable tree, it does not do anything but have one child.
- class executable_tree::Function

  Node that execute a function (that must be implemented as a child of Function class)
- class executable_tree::Value

  Node that has a static value.
- class executable_tree::Getter

  Node that get dynamic information in the frame.

Namespaces

- namespace executable_tree

7.5.1 Detailed Description

This file contains base class of a executable tree.

Author

Pagano Florian

Version

0.1

Date

2025

Copyright

7.6 node.hpp

Go to the documentation of this file.

```
1
11 #ifndef EXECUTABLE_TREE_HPP
12 #define EXECUTABLE_TREE_HPP
13
14 #include "decoder/frame.hpp"
15
16 #include <vector>
17 #include <string>
18
19 namespace executable_tree{
25     class Node {
26         public:
33             virtual std::string get_value(const decoder::Frame *target_frame) const;
34
40             virtual void add_node(Node* arg);
41
48             virtual std::string to_string(size_t depth) const;
49     };
50
56     class Root :  public Node {
57         private:
58             Node* _first_node = nullptr;
59
60         public:
65             ~Root(){
66                 if(_first_node)
67                     delete _first_node;
68             };
69
70             std::string get_value(const decoder::Frame *target_frame) const override;
71
72             void add_node(Node* arg) override;
73
74             std::string to_string(size_t depth) const override;
75     };
76
82     class Function :  public Node {
83         protected:
84             std::vector<Node*> args;
85
86         public:
91             ~Function(){
92                 for (Node* arg :  args)
93                     delete arg;
94
```

```
95                 args.clear();
96             }
97
98             void add_node(Node* arg) override;
99
100             std::string to_string(size_t depth) const override;
101     };
102
108     class Value :  public Node {
109         private:
110             std::string value;
111
112         public:
118             Value(std::string value) :  value(value) {};
119
120             std::string get_value(const decoder::Frame *target_frame) const override;
121
122             void add_node(Node* arg) override;
123
124             std::string to_string(size_t depth) const override;
125     };
126
132     class Getter :  public Node {
133         private:
134             const std::string field_name;
135
136         public:
142             Getter(const std::string field_name)
143                 :  field_name(field_name){};
144
145             std::string get_value(const decoder::Frame *target_frame) const override;
146
147             void add_node(Node* arg) override;
148
149             std::string to_string(size_t depth) const override;
150     };
151 }
152
153 #endif
```

7.7 includes/const.hpp File Reference

Contains all constant that are used in snapdesk.

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

Macros

- #define FRAME_HEADER_MIN_LENGTH 24

  The min length of the header of a frame.
- #define FRAME_HEADER_MAX_LENGTH 36

  The max length of the header of a frame.
- #define BEACON_FRAME_MAX_LENGTH 2346

  The max length of a beacon frame.
- #define BEACON_FRAME_BODY_MIN_LENGTH 12

  The min length of the body of a beacon frame.
- #define FRAME_MAX_LENGTH BEACON_FRAME_MAX_LENGTH

  The max length of a frame.
- #define DATABASE_ROOT "database"

  The root directory name where logs will be saved.

7.7.1 Detailed Description

Contains all constant that are used in snapdesk.

Author

Pagano Florian

Version

0.1

Date

2025

Copyright

7.7.2 Macro Definition Documentation

7.7.2.1 BEACON_FRAME_BODY_MIN_LENGTH

```
#define BEACON_FRAME_BODY_MIN_LENGTH 12
```

The min length of the body of a beacon frame.

7.7.2.2 BEACON_FRAME_MAX_LENGTH

```
#define BEACON_FRAME_MAX_LENGTH 2346
```

The max length of a beacon frame.

7.7.2.3 DATABASE_ROOT

```
#define DATABASE_ROOT "database"
```

The root directory name where logs will be saved.

7.7.2.4 FRAME_HEADER_MAX_LENGTH

```
#define FRAME_HEADER_MAX_LENGTH 36
```

The max length of the header of a frame.

7.7.2.5 FRAME_HEADER_MIN_LENGTH

```
#define FRAME_HEADER_MIN_LENGTH 24
```

The min length of the header of a frame.

7.7.2.6 FRAME_MAX_LENGTH

```
#define FRAME_MAX_LENGTH BEACON_FRAME_MAX_LENGTH
```

The max length of a frame.

7.8 const.hpp

Go to the documentation of this file.

```
1
12 #ifndef CONST_HPP
13 #define CONST_HPP
14
15 #define FRAME_HEADER_MIN_LENGTH 24
16 #define FRAME_HEADER_MAX_LENGTH 36
17
18 #define BEACON_FRAME_MAX_LENGTH 2346
19 #define BEACON_FRAME_BODY_MIN_LENGTH 12
20
21 // To change whith true value
22 #define FRAME_MAX_LENGTH BEACON_FRAME_MAX_LENGTH
23
24 #define DATABASE_ROOT "database"
25
26 #endif
```

7.9 includes/database/core.hpp File Reference

Contains core classes to manipulate the database.

```
#include <string>
#include <vector>
#include <algorithm>
#include <openssl/evp.h>
#include "os_communicator/os_communicator.hpp"
#include "const.hpp"
```

Include dependency graph for core.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class database::Csv

  This class represent and operates a csv file.
- class database::Database

  This class provide a simple API to manipulate storage files.

Namespaces

- namespace database

Macros

- #define HEADER_SIZE 1
- #define DELIMITER ";"
- #define DATA_EXTENTION ".csv"
- #define METADATA_EXTENTION ".metadata"
- #define KEY_LINE_NUMBER 0
- #define COLUMN_NUMBER_LINE_NUMBER 1
- #define COLUMN_NAMES {"output", "creation_date", "last_date"}
- #define KEY_COLUMN_NUMBER 0

7.9.1 Detailed Description

Contains core classes to manipulate the database.

Author

Pagano Florian

Version

0.1

Date

2025

Copyright

7.9.2 Macro Definition Documentation

7.9.2.1 COLUMN_NAMES

```
#define COLUMN_NAMES {"output", "creation_date", "last_date"}
```

7.9.2.2 COLUMN_NUMBER_LINE_NUMBER

```
#define COLUMN_NUMBER_LINE_NUMBER 1
```

7.9.2.3 DATA_EXTENTION

```
#define DATA_EXTENTION ".csv"
```

7.9.2.4 DELIMITER

```
#define DELIMITER ";"
```

7.9.2.5 HEADER_SIZE

```
#define HEADER_SIZE 1
```

7.9.2.6 KEY_COLUMN_NUMBER

```
#define KEY_COLUMN_NUMBER 0
```

7.9.2.7 KEY_LINE_NUMBER

```
#define KEY_LINE_NUMBER 0
```

7.9.2.8 METADATA_EXTENTION

```
#define METADATA_EXTENTION ".metadata"
```

7.10 core.hpp

Go to the documentation of this file.

```
1
11 #ifndef CORE_HPP
12 #define CORE_HPP
13
14 #include <string>
15 #include <vector>
16 #include <algorithm>
17
18 #include <openssl/evp.h>
19
20 #include "os_communicator/os_communicator.hpp"
21 #include "const.hpp"
22
23 // Number of line that are not raw data
24 #define HEADER_SIZE 1
25 #define DELIMITER ";"
26 #define DATA_EXTENTION ".csv"
27 #define METADATA_EXTENTION ".metadata"
28 #define KEY_LINE_NUMBER 0
29 #define COLUMN_NUMBER_LINE_NUMBER 1
30 #define COLUMN_NAMES {"output", "creation_date", "last_date"}
31 #define KEY_COLUMN_NUMBER 0
32
33 namespace database{
39     class Csv{
40         private:
41             size_t _number_of_columns;
42             size_t _key_column_number;
43             os_communicator::Communicator *_c_metadata;
44             os_communicator::Communicator *_c_data;
45
51             void _test_values_size(std::vector<std::string> values);
58             size_t _get_key_position(std::string key_value);
65             static std::string _values_to_line(std::vector<std::string> values);
73             size_t _get_column_position_from_header(std::string header, std::string column_name);
81             static std::string _get_cell_from_row(std::string row, size_t column_number);
88             size_t _string_to_size_t(std::string number);
89
90         public:
96             Csv(std::string file_name);
101             ~Csv();
110             static Csv *create(std::string file_name, std::vector<std::string> column_names, size_t
      key_column_number);
111
117             void push_row(std::vector<std::string> values);
124             void replace_row(size_t row_number, std::vector<std::string> values);
132             std::string get_cell(size_t row_number, size_t column_number);
139             size_t get_row_number(std::string key_value);
146             size_t get_column_number(std::string column_name);
152             std::vector<std::string> get_all_keys();
160             std::vector<std::string> get_all_keys_with_value(size_t column_number, std::string value);
161     };
162
168     class Database{
169         private:
170             std::string _ssid;
171             std::string _code_file_name;
172             std::string _code_hash;
173             Csv *_csv;
174
175         public:
182             Database(std::string ssid, std::string code_file_name);
187             ~Database();
188
196             std::string get_cell(std::string key_value, std::string column_name);
204             std::vector<std::string> get_key_of_entries(std::string column_name, std::string
      cell_value);
210             void add_entry(std::vector<std::string> values);
217             void replace_entry(std::string key_value, std::vector<std::string> values);
218     };
219 };
220
221 #endif
```

7.11 includes/decoder/big_number.hpp File Reference

```
#include <vector>
#include <cstdint>
```

```
#include <cstddef>
#include <stdexcept>
#include <string>
```

Include dependency graph for big_number.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class decoder::Big_number

  This class represent and manipulate indifined sized numbers using bytes.

Namespaces

- namespace decoder

7.11.1 Detailed Description

Author

Florian Pagano

Version

0.1

Date

2025

Copyright

Copyright (c) 2025

7.12 big_number.hpp

Go to the documentation of this file.

```
1
11 #ifndef BIG_NUMBER_HPP
12 #define BIG_NUMBER_HPP
13
14 #include <vector>
15 #include <cstdint>
16 #include <cstddef>
17 #include <stdexcept>
18 #include <string>
19
20
21 namespace decoder{
27     class Big_number {
28         private:
29             std::vector<uint8_t> number;
30             bool _is_null = false;
31
38             Big_number& operator=(const std::vector<uint8_t> _number);
39
40         public:
47             bool is_null() const;
52             void throw_if_null() const;
```

```
58            std::string hex_string() const;
64            std::string char_string() const;
70            size_t to_size_t() const;
77            void cut_byte(size_t from, size_t size);
84            void cut_bit(size_t from, size_t size);
90            Big_number copy() const;
91
92            /* Operators */
93
100            Big_number& operator=(const Big_number _number);
101
102            /* Constructors */
103
109            static Big_number null();
118            static Big_number from_buffer(const uint8_t *buffer, const size_t buffer_size, const size_t
    number_size);
127            static Big_number from_buffer_inv(const uint8_t *buffer, const size_t buffer_size, const
    size_t number_size);
134            static Big_number from_hex_string(const std::string string);
135    };
136 }
137
138 #endif
```

7.13 includes/decoder/frame.hpp File Reference

The file contains all basic frame class.

```
#include <cstdint>
#include <const.hpp>
#include <string>
#include <vector>
#include "os_communicator/os_communicator.hpp"
#include "decoder/big_number.hpp"
```

Include dependency graph for frame.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class decoder::Body

  This class set the base of all frame's body possible.
- class decoder::Frame

  The class that represent the whole frame.

Namespaces

- namespace decoder

7.13.1 Detailed Description

The file contains all basic frame class.

Author

Pagano Florian

Version

0.1

Date

2025

Copyright

7.14 frame.hpp

Go to the documentation of this file.

```
1
11 #ifndef FRAME_HPP
12 #define FRAME_HPP
13
14 #include <cstdint>
15 #include <const.hpp>
16 #include <string>
17 #include <vector>
18
19 #include "os_communicator/os_communicator.hpp"
20 #include "decoder/big_number.hpp"
21
22 using namespace std;
23
24 namespace decoder{
29     class Body {
30         protected:
31             uint8_t *_raw_body_buffer;
32             size_t _raw_buffer_size;
33
38             virtual void decode() = 0;
39
40         public:
47             Body(uint8_t *raw_body_buffer, size_t raw_buffer_size);
48
53             virtual void print() const = 0;
60             virtual Big_number get_value(string field) const = 0;
61
71             static Body *get_body(uint8_t *raw_body_buffer, size_t raw_buffer_size, size_t type, size_t
    sub_type);
72     };
73
79     class Frame{
80         private:
81             os_communicator::Communicator *communicator;
82         protected:
83             uint8_t *raw_frame_buffer;
84             size_t raw_buffer_size;
85             size_t raw_frame_size;
86             bool is_decoded;
87             bool has_raw_data;
88
89             // Header
90             Big_number frame_control = Big_number::null();
91             Big_number duration = Big_number::null();
92             Big_number destination_address = Big_number::null();
93             Big_number source_address = Big_number::null();
94             Big_number bssid = Big_number::null();
95             Big_number sequence_control = Big_number::null();
96             Big_number frame_check_sum = Big_number::null();
97
98             // Body
99             Body *body = nullptr;
100
101         public:
107             Frame(os_communicator::Communicator *_communicator);
112             ~Frame();
113
120             bool get_is_decoded() const;
125             void update_raw_data();
```

```
130             void print_raw_data();
135             virtual void decode();
140             virtual void print() const;
147             Big_number get_value(string field) const;
148     };
149 }
150
151 #endif
```

7.15 includes/decoder/ie.hpp File Reference

Classes that permit to represent the IE of a frame's body.

```
#include <cstdint>
#include <stdexcept>
#include <map>
#include "decoder/big_number.hpp"
```

Include dependency graph for ie.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class decoder::Ie_node

  An element of a linked list of IEs.

Namespaces

- namespace decoder

Macros

- #define P_SSID 0

  element id 0 is the SSID
- #define P_SUPPORTED_RATES 1

  element id 1 is the supported rated
- #define P_FH 2

  element id 2 is the FH parameter set
- #define P_DS 3

  element id 3 is the DS parameter set
- #define P_CF 4

  element id 4 is the CF parameter set
- #define P_TIM 5

  element id 5 is the TIM
- #define P_IBSS 6

  element id 6 is the IBSS
- #define P_Challenge_text 16

  element 16 is the challenge text

7.15.1 Detailed Description

Classes that permit to represent the IE of a frame's body.

Author

Pagano Florian

Version

0.1

Date

2025

Copyright

7.15.2 Macro Definition Documentation

7.15.2.1 P_CF

```
#define P_CF 4
```

element id 4 is the CF parameter set

7.15.2.2 P_Challenge_text

```
#define P_Challenge_text 16
```

element 16 is the challenge text

7.15.2.3 P_DS

```
#define P_DS 3
```

element id 3 is the DS parameter set

7.15.2.4 P_FH

```
#define P_FH 2
```

element id 2 is the FH parameter set

7.15.2.5 P_IBSS

```
#define P_IBSS 6
```

element id 6 is the IBSS

7.15.2.6 P_SSID

```
#define P_SSID 0
```

element id 0 is the SSID

7.15.2.7 P_SUPPORTED_RATES

```
#define P_SUPPORTED_RATES 1
```

element id 1 is the supported rated

7.15.2.8 P_TIM

```
#define P_TIM 5
```

element id 5 is the TIM

7.16 ie.hpp

Go to the documentation of this file.

```
1
11 #ifndef BEACON_ELEMENT_HPP
12 #define BEACON_ELEMENT_HPP
13
14 #include <cstdint>
15 #include <stdexcept>
16 #include <stdexcept>
17 #include <map>
18 #include "decoder/big_number.hpp"
19
20 #define P_SSID 0
21 #define P_SUPPORTED_RATES 1
22 #define P_FH 2
23 #define P_DS 3
24 #define P_CF 4
25 #define P_TIM 5
26 #define P_IBSS 6
27 #define P_Challenge_text 16
28
29 namespace decoder{
35     class Ie_node {
36         private:
37             uint8_t element_id;
38             Big_number element_value;
39             Ie_node *next_element;
40
48             bool is_valid_hex(const std::string str) const;
49
50         public:
57             Ie_node(uint8_t element_id, Big_number value);
62             ~Ie_node();
63
69             void add(Ie_node *new_element);
76             Big_number get_value(std::string field) const;
81             void print();
82     };
83 }
84
85 #endif
```

7.17 includes/decoder/management_body.hpp File Reference

```
#include <cstdint>
#include <const.hpp>
#include <string>
#include "decoder/frame.hpp"
#include "decoder/ie.hpp"
#include "decoder/big_number.hpp"
```

Include dependency graph for management_body.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class decoder::Beacon_body

  Is the body of a beacon frame.

Namespaces

- namespace decoder

7.17.1 Detailed Description

Author

Pagano Florian

Version

0.1

Date

2025

Copyright

7.18 management_body.hpp

Go to the documentation of this file.

```
1
11 #ifndef MANAGEMENT_BODY_HPP
12 #define MANAGEMENT_BODY_HPP
13
14 #include <cstdint>
15 #include <const.hpp>
16 #include <string>
17 #include "decoder/frame.hpp"
18 #include "decoder/ie.hpp"
19 #include "decoder/big_number.hpp"
20
21 using namespace std;
22
23 namespace decoder{
29     class Beacon_body :  public Body {
30         private:
31             Big_number _timestamp;
32             Big_number _beacon_interval;
33             Big_number _capabilities_information;
34             Ie_node *_first_ie = nullptr;
35
40             void decode() override;
48             void add_ie(uint8_t element_id, uint8_t element_length, size_t start_position);
49
50         public:
57             Beacon_body(uint8_t *raw_body_buffer, size_t raw_buffer_size);
62             ~Beacon_body();
63
68             void print() const override;
75             Big_number get_value(string field) const override;
76     };
77 }
78
79 #endif
```

7.19 includes/os_communicator/os_communicator.hpp File Reference

The os_communicator component handle all interaction between snapdesk and the current os.

```
#include <cstddef>
#include <cstdint>
#include <stdexcept>
#include <string>
#include <fstream>
#include <ctime>
#include <iomanip>
#include <chrono>
#include <thread>
#include <filesystem>
```

Include dependency graph for os_communicator.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class os_communicator::Communicator

  The Communicator is the main class of the os_communicator component.

Namespaces

- namespace os_communicator

Macros

- #define F_NONE 0

  A file that does not exist.
- #define F_FILE 1

  A file.
- #define F_FOLDER 2

  A folder.

7.19.1 Detailed Description

The os_communicator component handle all interaction between snapdesk and the current os.

Author

Pagano Florian

Version

0.1

Date

2025

Copyright

7.19.2 Macro Definition Documentation

7.19.2.1 F_FILE

```
#define F_FILE 1
```

A file.

7.19.2.2 F_FOLDER

```
#define F_FOLDER 2
```

A folder.

7.19.2.3 F_NONE

```
#define F_NONE 0
```

A file that does not exist.

7.20 os_communicator.hpp

Go to the documentation of this file.

```
1
12 #ifndef OS_COMMUNICATOR_HPP
13 #define OS_COMMUNICATOR_HPP
14
15 #include <cstddef>
16 #include <cstdint>
17 #include <stdexcept>
18 #include <string>
19 #include <fstream>
20 #include <ctime>
21 #include <iomanip>
22 #include <chrono>
23 #include <thread>
24 #include <filesystem>
25
26 #define F_NONE 0
27 #define F_FILE 1
28 #define F_FOLDER 2
29
30 using namespace std;
31
32 namespace os_communicator{
33
39     class Communicator{
40         private:
41             string _file_name;
42
43         public:
49             Communicator(string file_name);
50
57             bool exist();
65             size_t in_buffer(uint8_t *buffer, const size_t buffer_size);
```

```
72              string get_line(size_t line_number) const;
78              void add_line(string line);
85              void replace_line(size_t line_number, string new_line);
90              void new_file();
91
97              static string get_current_date();
103              static void notify(string message);
109              static void sleep(size_t seconds);
110
116              static void create_folder(string folder_name);
123              static size_t get_file_type(string name);
124     };
125 }
126
127 #endif
```

7.21 src/compiler/compiler.cpp File Reference

`#include "compiler/compiler.hpp"`
Include dependency graph for compiler.cpp:

7.22 src/compiler/function_node.cpp File Reference

`#include "compiler/function_node.hpp"`
Include dependency graph for function_node.cpp:

7.23 src/compiler/node.cpp File Reference

`#include "compiler/node.hpp"`
Include dependency graph for node.cpp:

7.24 src/database/csv.cpp File Reference

`#include "database/core.hpp"`
Include dependency graph for csv.cpp:

Namespaces

- namespace database

7.25 src/database/database.cpp File Reference

`#include "database/core.hpp"`
Include dependency graph for database.cpp:

Namespaces

- namespace database

Functions

- std::string database::hash (std::string str)

7.26 src/decoder/big_number.cpp File Reference

```
#include "decoder/big_number.hpp"
```
Include dependency graph for big_number.cpp:

7.27 src/decoder/frame.cpp File Reference

```
#include "decoder/frame.hpp"
#include "decoder/management_body.hpp"
```
Include dependency graph for frame.cpp:

7.28 src/decoder/ie.cpp File Reference

```
#include "decoder/ie.hpp"
```
Include dependency graph for ie.cpp:

7.29 src/decoder/management_body.cpp File Reference

```
#include "decoder/management_body.hpp"
```
Include dependency graph for management_body.cpp:

7.30 src/main.cpp File Reference

This is the main file. It contains the main loop of snapdesk.

```
#include "decoder/frame.hpp"
#include "os_communicator/os_communicator.hpp"
#include "compiler/node.hpp"
#include "compiler/function_node.hpp"
#include "compiler/compiler.hpp"
#include "database/core.hpp"
```
Include dependency graph for main.cpp:

Macros

- #define CHARACTER_DEVICE_FILE "/dev/beacon-sniffer-0"
- #define SCRIPT_FILE "./snappy.txt"
- #define PERIOD 1

Functions

- int run ()

  The true main function. This exist to permit the program to rerun itself when crashing.
- int main (int argc, char ∗argv[ ])

7.30.1 Detailed Description

This is the main file. It contains the main loop of snapdesk.

Author

Pagano Florian

Version

0.1

Date

2025-05-15

Copyright

Copyright (c) 2025

7.30.2 Macro Definition Documentation

7.30.2.1 CHARACTER_DEVICE_FILE

```
#define CHARACTER_DEVICE_FILE "/dev/beacon-sniffer-0"
```

7.30.2.2 PERIOD

```
#define PERIOD 1
```

7.30.2.3 SCRIPT_FILE

```
#define SCRIPT_FILE "./snappy.txt"
```

7.30.3 Function Documentation

7.30.3.1 main()

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

7.30.3.2 run()

```
int run ( )
```

The true main function. This exist to permit the program to rerun itself when crashing.

Returns

int: The same return than the main function

7.31 src/os_communicator/os_communicator.cpp File Reference

```
#include "os_communicator/os_communicator.hpp"
```
Include dependency graph for os_communicator.cpp:

Namespaces

- namespace os_communicator

Index