

Snapdesk

Generated by Doxygen 1.9.4



|  |           |
|--|-----------|
| <b>1 Namespace Index</b>                     | <b>1</b>  |
| 1.1 Namespace List                           | 1         |
| <b>2 Hierarchical Index</b>                  | <b>3</b>  |
| 2.1 Class Hierarchy                          | 3         |
| <b>3 Class Index</b>                         | <b>5</b>  |
| 3.1 Class List                               | 5         |
| <b>4 File Index</b>                          | <b>7</b>  |
| 4.1 File List                                | 7         |
| <b>5 Namespace Documentation</b>             | <b>9</b>  |
| 5.1 compiler Namespace Reference             | 9         |
| 5.2 database Namespace Reference             | 9         |
| 5.2.1 Function Documentation                 | 9         |
| 5.2.1.1 hash()                               | 9         |
| 5.3 decoder Namespace Reference              | 10        |
| 5.4 executable_tree Namespace Reference      | 10        |
| 5.5 os_communicator Namespace Reference      | 10        |
| <b>6 Class Documentation</b>                 | <b>11</b> |
| 6.1 decoder::Beacon_body Class Reference     | 11        |
| 6.1.1 Detailed Description                   | 11        |
| 6.1.2 Constructor & Destructor Documentation | 11        |
| 6.1.2.1 Beacon_body()                        | 11        |
| 6.1.2.2 ~Beacon_body()                       | 12        |
| 6.1.3 Member Function Documentation          | 12        |
| 6.1.3.1 get_value()                          | 12        |
| 6.1.3.2 print()                              | 12        |
| 6.2 decoder::Big_number Class Reference      | 13        |
| 6.2.1 Detailed Description                   | 13        |
| 6.2.2 Member Function Documentation          | 13        |
| 6.2.2.1 char_string()                        | 14        |
| 6.2.2.2 copy()                               | 14        |
| 6.2.2.3 cut_bit()                            | 14        |
| 6.2.2.4 cut_byte()                           | 14        |
| 6.2.2.5 from_buffer()                        | 15        |
| 6.2.2.6 from_buffer_inv()                    | 15        |
| 6.2.2.7 from_hex_string()                    | 16        |
| 6.2.2.8 hex_string()                         | 16        |
| 6.2.2.9 is_null()                            | 16        |
| 6.2.2.10 null()                              | 16        |
| 6.2.2.11 operator=()                         | 16        |

|   |    |
|---|----|
| 6.2.2.12 throw_if_null()                          | 17 |
| 6.2.2.13 to_size_t()                              | 17 |
| 6.3 decoder::Body Class Reference                 | 17 |
| 6.3.1 Detailed Description                        | 18 |
| 6.3.2 Constructor & Destructor Documentation      | 18 |
| 6.3.2.1 Body()                                    | 18 |
| 6.3.3 Member Function Documentation               | 18 |
| 6.3.3.1 decode()                                  | 19 |
| 6.3.3.2 get_body()                                | 19 |
| 6.3.3.3 get_value()                               | 19 |
| 6.3.3.4 print()                                   | 20 |
| 6.3.4 Member Data Documentation                   | 20 |
| 6.3.4.1 _raw_body_buffer                          | 20 |
| 6.3.4.2 _raw_buffer_size                          | 20 |
| 6.4 os_communicator::Communicator Class Reference | 20 |
| 6.4.1 Detailed Description                        | 21 |
| 6.4.2 Constructor & Destructor Documentation      | 21 |
| 6.4.2.1 Communicator()                            | 21 |
| 6.4.3 Member Function Documentation               | 21 |
| 6.4.3.1 add_line()                                | 21 |
| 6.4.3.2 create_folder()                           | 22 |
| 6.4.3.3 exist()                                   | 22 |
| 6.4.3.4 get_current_date()                        | 22 |
| 6.4.3.5 get_file_type()                           | 22 |
| 6.4.3.6 get_line()                                | 23 |
| 6.4.3.7 in_buffer()                               | 23 |
| 6.4.3.8 new_file()                                | 24 |
| 6.4.3.9 notify()                                  | 24 |
| 6.4.3.10 replace_line()                           | 24 |
| 6.4.3.11 sleep()                                  | 24 |
| 6.5 compiler::Compiler Class Reference            | 25 |
| 6.5.1 Detailed Description                        | 25 |
| 6.5.2 Constructor & Destructor Documentation      | 25 |
| 6.5.2.1 Compiler()                                | 25 |
| 6.5.3 Member Function Documentation               | 25 |
| 6.5.3.1 get_executable_tree()                     | 25 |
| 6.6 database::Csv Class Reference                 | 26 |
| 6.6.1 Detailed Description                        | 26 |
| 6.6.2 Constructor & Destructor Documentation      | 27 |
| 6.6.2.1 Csv()                                     | 27 |
| 6.6.2.2 ~Csv()                                    | 28 |
| 6.6.3 Member Function Documentation               | 28 |

|   |    |
|---|----|
| 6.6.3.1 create()                              | 28 |
| 6.6.3.2 get_all_keys()                        | 28 |
| 6.6.3.3 get_all_keys_with_value()             | 29 |
| 6.6.3.4 get_cell()                            | 29 |
| 6.6.3.5 get_column_number()                   | 29 |
| 6.6.3.6 get_row_number()                      | 30 |
| 6.6.3.7 push_row()                            | 30 |
| 6.6.3.8 replace_row()                         | 30 |
| 6.7 executable_tree::Cut_bit Class Reference  | 31 |
| 6.7.1 Detailed Description                    | 31 |
| 6.7.2 Member Function Documentation           | 31 |
| 6.7.2.1 get_value()                           | 31 |
| 6.7.2.2 to_string()                           | 32 |
| 6.8 executable_tree::Cut_byte Class Reference | 32 |
| 6.8.1 Detailed Description                    | 33 |
| 6.8.2 Member Function Documentation           | 33 |
| 6.8.2.1 get_value()                           | 33 |
| 6.8.2.2 to_string()                           | 33 |
| 6.9 database::Database Class Reference        | 34 |
| 6.9.1 Detailed Description                    | 34 |
| 6.9.2 Constructor & Destructor Documentation  | 34 |
| 6.9.2.1 Database()                            | 34 |
| 6.9.2.2 ~Database()                           | 35 |
| 6.9.3 Member Function Documentation           | 35 |
| 6.9.3.1 add_entry()                           | 35 |
| 6.9.3.2 get_cell()                            | 35 |
| 6.9.3.3 get_key_of_entries()                  | 36 |
| 6.9.3.4 replace_entry()                       | 36 |
| 6.10 decoder::Frame Class Reference           | 36 |
| 6.10.1 Detailed Description                   | 37 |
| 6.10.2 Constructor & Destructor Documentation | 37 |
| 6.10.2.1 Frame()                              | 38 |
| 6.10.2.2 ~Frame()                             | 39 |
| 6.10.3 Member Function Documentation          | 39 |
| 6.10.3.1 decode()                             | 39 |
| 6.10.3.2 get_is_decoded()                     | 39 |
| 6.10.3.3 get_value()                          | 39 |
| 6.10.3.4 print()                              | 40 |
| 6.10.3.5 print_raw_data()                     | 40 |
| 6.10.3.6 update_raw_data()                    | 40 |
| 6.10.4 Member Data Documentation              | 40 |
| 6.10.4.1 body                                 | 40 |

|  |    |
|--|----|
| 6.10.4.2 bssid . . . . .                                 | 41 |
| 6.10.4.3 destination_address . . . . .                   | 41 |
| 6.10.4.4 duration . . . . .                              | 41 |
| 6.10.4.5 frame_check_sum . . . . .                       | 41 |
| 6.10.4.6 frame_control . . . . .                         | 41 |
| 6.10.4.7 has_raw_data . . . . .                          | 41 |
| 6.10.4.8 is_decoded . . . . .                            | 41 |
| 6.10.4.9 raw_buffer_size . . . . .                       | 42 |
| 6.10.4.10 raw_frame_buffer . . . . .                     | 42 |
| 6.10.4.11 raw_frame_size . . . . .                       | 42 |
| 6.10.4.12 sequence_control . . . . .                     | 42 |
| 6.10.4.13 source_address . . . . .                       | 42 |
| 6.11 executable_tree::Function Class Reference . . . . . | 42 |
| 6.11.1 Detailed Description . . . . .                    | 43 |
| 6.11.2 Constructor & Destructor Documentation . . . . .  | 43 |
| 6.11.2.1 ~Function() . . . . .                           | 43 |
| 6.11.3 Member Function Documentation . . . . .           | 43 |
| 6.11.3.1 add_node() . . . . .                            | 43 |
| 6.11.3.2 to_string() . . . . .                           | 44 |
| 6.11.4 Member Data Documentation . . . . .               | 44 |
| 6.11.4.1 args . . . . .                                  | 44 |
| 6.12 executable_tree::Getter Class Reference . . . . .   | 44 |
| 6.12.1 Detailed Description . . . . .                    | 45 |
| 6.12.2 Constructor & Destructor Documentation . . . . .  | 45 |
| 6.12.2.1 Getter() . . . . .                              | 45 |
| 6.12.3 Member Function Documentation . . . . .           | 45 |
| 6.12.3.1 add_node() . . . . .                            | 45 |
| 6.12.3.2 get_value() . . . . .                           | 46 |
| 6.12.3.3 to_string() . . . . .                           | 46 |
| 6.13 decoder::le_node Class Reference . . . . .          | 46 |
| 6.13.1 Detailed Description . . . . .                    | 47 |
| 6.13.2 Constructor & Destructor Documentation . . . . .  | 47 |
| 6.13.2.1 le_node() . . . . .                             | 47 |
| 6.13.2.2 ~le_node() . . . . .                            | 47 |
| 6.13.3 Member Function Documentation . . . . .           | 47 |
| 6.13.3.1 add() . . . . .                                 | 48 |
| 6.13.3.2 get_value() . . . . .                           | 48 |
| 6.13.3.3 print() . . . . .                               | 48 |
| 6.14 executable_tree::Node Class Reference . . . . .     | 48 |
| 6.14.1 Detailed Description . . . . .                    | 49 |
| 6.14.2 Member Function Documentation . . . . .           | 49 |
| 6.14.2.1 add_node() . . . . .                            | 49 |

|   |           |
|---|-----------|
| 6.14.2.2 <code>get_value()</code> . . . . .                                   | 49        |
| 6.14.2.3 <code>to_string()</code> . . . . .                                   | 50        |
| 6.15 <code>executable_tree::Root</code> Class Reference . . . . .             | 50        |
| 6.15.1 Detailed Description . . . . .   | 51        |
| 6.15.2 Constructor & Destructor Documentation . . . . .                       | 51        |
| 6.15.2.1 <code>~Root()</code> . . . . .                                       | 51        |
| 6.15.3 Member Function Documentation . . . . .                                | 51        |
| 6.15.3.1 <code>add_node()</code> . . . . .                                    | 51        |
| 6.15.3.2 <code>get_value()</code> . . . . .                                   | 51        |
| 6.15.3.3 <code>to_string()</code> . . . . .                                   | 52        |
| 6.16 <code>executable_tree::Sha256</code> Class Reference . . . . .           | 52        |
| 6.16.1 Detailed Description . . . . .   | 53        |
| 6.16.2 Member Function Documentation . . . . .                                | 53        |
| 6.16.2.1 <code>get_value()</code> . . . . .                                   | 53        |
| 6.16.2.2 <code>to_string()</code> . . . . .                                   | 53        |
| 6.17 <code>executable_tree::Value</code> Class Reference . . . . .            | 54        |
| 6.17.1 Detailed Description . . . . .   | 54        |
| 6.17.2 Constructor & Destructor Documentation . . . . .                       | 54        |
| 6.17.2.1 <code>Value()</code> . . . . .                                       | 54        |
| 6.17.3 Member Function Documentation . . . . .                                | 55        |
| 6.17.3.1 <code>add_node()</code> . . . . .                                    | 55        |
| 6.17.3.2 <code>get_value()</code> . . . . .                                   | 55        |
| 6.17.3.3 <code>to_string()</code> . . . . .                                   | 55        |
| <b>7 File Documentation</b> . . . . .   | <b>57</b> |
| 7.1 <code>includes/compiler/compiler.hpp</code> File Reference . . . . .      | 57        |
| 7.2 <code>compiler.hpp</code> . . . . .                                       | 57        |
| 7.3 <code>includes/compiler/function_node.hpp</code> File Reference . . . . . | 58        |
| 7.3.1 Detailed Description . . . . .  | 58        |
| 7.4 <code>function_node.hpp</code> . . . . .                                  | 59        |
| 7.5 <code>includes/compiler/node.hpp</code> File Reference . . . . .          | 59        |
| 7.5.1 Detailed Description . . . . .  | 60        |
| 7.6 <code>node.hpp</code> . . . . .   | 60        |
| 7.7 <code>includes/const.hpp</code> File Reference . . . . .                  | 61        |
| 7.7.1 Detailed Description . . . . .  | 62        |
| 7.7.2 Macro Definition Documentation . . . . .                                | 62        |
| 7.7.2.1 <code>BEACON_FRAME_BODY_MIN_LENGTH</code> . . . . .                   | 62        |
| 7.7.2.2 <code>BEACON_FRAME_MAX_LENGTH</code> . . . . .                        | 62        |
| 7.7.2.3 <code>DATABASE_ROOT</code> . . . . .                                  | 62        |
| 7.7.2.4 <code>FRAME_HEADER_MAX_LENGTH</code> . . . . .                        | 63        |
| 7.7.2.5 <code>FRAME_HEADER_MIN_LENGTH</code> . . . . .                        | 63        |
| 7.7.2.6 <code>FRAME_MAX_LENGTH</code> . . . . .                               | 63        |

|  |    |
|--|----|
| 7.8 const.hpp  | 63 |
| 7.9 includes/database/core.hpp File Reference                    | 63 |
| 7.9.1 Detailed Description                                       | 64 |
| 7.9.2 Macro Definition Documentation                             | 64 |
| 7.9.2.1 COLUMN_NAMES   | 65 |
| 7.9.2.2 COLUMN_NUMBER_LINE_NUMBER                                | 65 |
| 7.9.2.3 DATA_EXTENTION   | 65 |
| 7.9.2.4 DELIMITER  | 65 |
| 7.9.2.5 HEADER_SIZE  | 65 |
| 7.9.2.6 KEY_COLUMN_NUMBER  | 65 |
| 7.9.2.7 KEY_LINE_NUMBER  | 65 |
| 7.9.2.8 METADATA_EXTENTION                                       | 65 |
| 7.10 core.hpp  | 66 |
| 7.11 includes/decoder/big_number.hpp File Reference              | 66 |
| 7.11.1 Detailed Description                                      | 67 |
| 7.12 big_number.hpp  | 67 |
| 7.13 includes/decoder/frame.hpp File Reference                   | 68 |
| 7.13.1 Detailed Description                                      | 68 |
| 7.14 frame.hpp   | 69 |
| 7.15 includes/decoder/ie.hpp File Reference                      | 70 |
| 7.15.1 Detailed Description                                      | 71 |
| 7.15.2 Macro Definition Documentation                            | 71 |
| 7.15.2.1 P_CF  | 71 |
| 7.15.2.2 P_Challenge_text  | 71 |
| 7.15.2.3 P_DS  | 71 |
| 7.15.2.4 P_FH  | 72 |
| 7.15.2.5 P_IBSS  | 72 |
| 7.15.2.6 P_SSID  | 72 |
| 7.15.2.7 P_SUPPORTED_RATES                                       | 72 |
| 7.15.2.8 P_TIM   | 72 |
| 7.16 ie.hpp  | 73 |
| 7.17 includes/decoder/management_body.hpp File Reference         | 73 |
| 7.17.1 Detailed Description                                      | 74 |
| 7.18 management_body.hpp   | 74 |
| 7.19 includes/os_communicator/os_communicator.hpp File Reference | 75 |
| 7.19.1 Detailed Description                                      | 75 |
| 7.19.2 Macro Definition Documentation                            | 76 |
| 7.19.2.1 F_FILE  | 76 |
| 7.19.2.2 F_FOLDER  | 76 |
| 7.19.2.3 F_NONE  | 76 |
| 7.20 os_communicator.hpp   | 76 |
| 7.21 src/compiler/compiler.cpp File Reference                    | 77 |



|   |           |
|---|-----------|
| 7.22 src/compiler/function_node.cpp File Reference . . . . .          | 77        |
| 7.23 src/compiler/node.cpp File Reference . . . . .                   | 77        |
| 7.24 src/database/csv.cpp File Reference . . . . .                    | 77        |
| 7.25 src/database/database.cpp File Reference . . . . .               | 77        |
| 7.26 src/decoder/big_number.cpp File Reference . . . . .              | 78        |
| 7.27 src/decoder/frame.cpp File Reference . . . . .                   | 78        |
| 7.28 src/decoder/ie.cpp File Reference . . . . .                      | 78        |
| 7.29 src/decoder/management_body.cpp File Reference . . . . .         | 78        |
| 7.30 src/main.cpp File Reference . . . . .                            | 78        |
| 7.30.1 Detailed Description . . . . .                                 | 79        |
| 7.30.2 Macro Definition Documentation . . . . .                       | 79        |
| 7.30.2.1 CHARACTER_DEVICE_FILE . . . . .                              | 79        |
| 7.30.2.2 PERIOD . . . . .   | 79        |
| 7.30.2.3 SCRIPT_FILE . . . . .  | 79        |
| 7.30.3 Function Documentation . . . . .                               | 80        |
| 7.30.3.1 main() . . . . .   | 80        |
| 7.30.3.2 run() . . . . .  | 80        |
| 7.31 src/os_communicator/os_communicator.cpp File Reference . . . . . | 80        |
| <b>Index</b>  | <b>81</b> |



# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

|                                 |    |
|---------------------------------|----|
| <a href="#">compiler</a>        | 9  |
| <a href="#">database</a>        | 9  |
| <a href="#">decoder</a>         | 10 |
| <a href="#">executable_tree</a> | 10 |
| <a href="#">os_communicator</a> | 10 |



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

|   |    |
|---|----|
| decoder::Big_number . . . . .           | 13 |
| decoder::Body . . . . .                 | 17 |
| decoder::Beacon_body . . . . .          | 11 |
| os_communicator::Communicator . . . . . | 20 |
| compiler::Compiler . . . . .            | 25 |
| database::Csv . . . . .                 | 26 |
| database::Database . . . . .            | 34 |
| decoder::Frame . . . . .                | 36 |
| decoder::le_node . . . . .              | 46 |
| executable_tree::Node . . . . .         | 48 |
| executable_tree::Function . . . . .     | 42 |
| executable_tree::Cut_bit . . . . .      | 31 |
| executable_tree::Cut_byte . . . . .     | 32 |
| executable_tree::Sha256 . . . . .       | 52 |
| executable_tree::Getter . . . . .       | 44 |
| executable_tree::Root . . . . .         | 50 |
| executable_tree::Value . . . . .        | 54 |



## Chapter 3

# Class Index

### 3.1 Class List

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

|  |    |
|--|----|
| <a href="#">decoder::Beacon_body</a>   |    |
| Is the body of a beacon frame . . . . .  | 11 |
| <a href="#">decoder::Big_number</a>  |    |
| This class represent and manipulate indifined sized numbers using bytes . . . . .  | 13 |
| <a href="#">decoder::Body</a>  |    |
| This class set the base of all frame's body possible . . . . .   | 17 |
| <a href="#">os_communicator::Communicator</a>  |    |
| The <a href="#">Communicator</a> is the main class of the <a href="#">os_communicator</a> component . . . . .                  | 20 |
| <a href="#">compiler::Compiler</a>   |    |
| Tranform the source code into a executable tree . . . . .  | 25 |
| <a href="#">database::Csv</a>  |    |
| This class represent and operates a csv file . . . . .   | 26 |
| <a href="#">executable_tree::Cut_bit</a>   |    |
| This <a href="#">Function</a> cut the bits of the first arg from the second arg of the size of the third arg . . . . .         | 31 |
| <a href="#">executable_tree::Cut_byte</a>  |    |
| This <a href="#">Function</a> cut the bytes of the first arg fril the secong arg of the size of the third arg . . . . .        | 32 |
| <a href="#">database::Database</a>   |    |
| This class provide a simple API to manipulate storage files . . . . .  | 34 |
| <a href="#">decoder::Frame</a>   |    |
| The class that represent the whole frame . . . . .   | 36 |
| <a href="#">executable_tree::Function</a>  |    |
| <a href="#">Node</a> that execute a function (that must be implemented as a child of <a href="#">Function</a> class) . . . . . | 42 |
| <a href="#">executable_tree::Getter</a>  |    |
| <a href="#">Node</a> that get dynamic information in the frame . . . . .   | 44 |
| <a href="#">decoder::le_node</a>   |    |
| An element of a linked list of IEs . . . . .   | 46 |
| <a href="#">executable_tree::Node</a>  |    |
| Represents a node of the executable tree . . . . .   | 48 |
| <a href="#">executable_tree::Root</a>  |    |
| The node that represent the root of the executable tree, it does not do anything but have one child . . . . .                  | 50 |
| <a href="#">executable_tree::Sha256</a>  |    |
| This <a href="#">Function</a> concatenates its arguments and . . . . .   | 52 |
| <a href="#">executable_tree::Value</a>   |    |
| <a href="#">Node</a> that has a static value . . . . .   | 54 |





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

|  |    |
|--|----|
| includes/ <a href="#">const.hpp</a>  | 61 |
| Contains all constant that are used in snapdesk . . . . .  |    |
| includes/compiler/ <a href="#">compiler.hpp</a> . . . . .  | 57 |
| includes/compiler/ <a href="#">function_node.hpp</a> . . . . .   | 58 |
| includes/compiler/ <a href="#">node.hpp</a>  |    |
| This file contains base class of a executable tree . . . . .   | 59 |
| includes/database/ <a href="#">core.hpp</a>  |    |
| Contains core classes to manipulate the database . . . . .   | 63 |
| includes/decoder/ <a href="#">big_number.hpp</a> . . . . .   | 66 |
| includes/decoder/ <a href="#">frame.hpp</a>  |    |
| The file contains all basic frame class . . . . .  | 68 |
| includes/decoder/ <a href="#">ie.hpp</a>   |    |
| Classes that permit to represent the IE of a frame's body . . . . .                                      | 70 |
| includes/decoder/ <a href="#">management_body.hpp</a> . . . . .  | 73 |
| includes/os_communicator/ <a href="#">os_communicator.hpp</a>  |    |
| The <a href="#">os_communicator</a> component handle all interaction between snapdesk and the current os | 75 |
| src/ <a href="#">main.cpp</a>  |    |
| This is the main file. It contains the main loop of snapdesk . . . . .                                   | 78 |
| src/compiler/ <a href="#">compiler.cpp</a> . . . . .   | 77 |
| src/compiler/ <a href="#">function_node.cpp</a> . . . . .  | 77 |
| src/compiler/ <a href="#">node.cpp</a> . . . . .   | 77 |
| src/database/ <a href="#">csv.cpp</a> . . . . .  | 77 |
| src/database/ <a href="#">database.cpp</a> . . . . .   | 77 |
| src/decoder/ <a href="#">big_number.cpp</a> . . . . .  | 78 |
| src/decoder/ <a href="#">frame.cpp</a> . . . . .   | 78 |
| src/decoder/ <a href="#">ie.cpp</a> . . . . .  | 78 |
| src/decoder/ <a href="#">management_body.cpp</a> . . . . .   | 78 |
| src/os_communicator/ <a href="#">os_communicator.cpp</a> . . . . .                                       | 80 |



## 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 [le\\_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

|                        |   |
|------------------------|---|
| <i>raw_body_buffer</i> | a buffer containing the exact beacon body |
| <i>raw_buffer_size</i> | 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

|              |   |
|--------------|---|
| <i>field</i> | 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

|             |                                 |
|-------------|---------------------------------|
| <i>from</i> | the first bit of the new number |
| <i>size</i> | 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

|             |                                  |
|-------------|----------------------------------|
| <i>from</i> | the first byte of the new number |
| <i>size</i> | 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

|                    |   |
|--------------------|---|
| <i>buffer</i>      | the buffer containing the number, starting at the first number byte |
| <i>buffer_size</i> | the size of the buffer  |
| <i>number_size</i> | 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

|                    |   |
|--------------------|---|
| <i>buffer</i>      | the buffer containing the number, starting at the first number byte |
| <i>buffer_size</i> | the size of the buffer  |
| <i>number_size</i> | 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

|               |   |
|---------------|---|
| <i>string</i> | 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

|                      |   |
|----------------------|---|
| <code>_number</code> | the <a href="#">Big_number</a> 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

|                        |                                    |
|------------------------|------------------------------------|
| <i>raw_body_buffer</i> | a buffer containing the exact body |
| <i>raw_buffer_size</i> | 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

|                        |  |
|------------------------|--|
| <i>raw_body_buffer</i> | a buffer containing the exact body                   |
| <i>raw_buffer_size</i> | the size of the buffer                               |
| <i>type</i>            | the type of the frame (management, control, or data) |
| <i>sub_type</i>        | 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

|              |  |
|--------------|--|
| <i>field</i> | 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

|                  |                                       |
|------------------|---------------------------------------|
| <i>file_name</i> | 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

|             |  |
|-------------|--|
| <i>line</i> | 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

|                    |                        |
|--------------------|------------------------|
| <i>folder_name</i> | 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**

|             |                      |
|-------------|----------------------|
| <i>name</i> | 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**

|                    |   |
|--------------------|---|
| <i>line_number</i> | 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**

|                    |  |
|--------------------|--|
| <i>buffer</i>      | the buffer where the data will be copied |
| <i>buffer_size</i> | 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

|                |                                    |
|----------------|------------------------------------|
| <i>message</i> | 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

|                    |  |
|--------------------|--|
| <i>line_number</i> | the number of the file to replace      |
| <i>new_line</i>    | 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

|                |  |
|----------------|--|
| <i>seconds</i> | 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

|                     |                             |
|---------------------|-----------------------------|
| <i>communicator</i> | 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

|                  |   |
|------------------|---|
| <i>file_name</i> | 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

|                          |   |
|--------------------------|---|
| <i>file_name</i>         | the filename of the new storage file (without extentions) |
| <i>column_names</i>      | the names of the columns                                  |
| <i>key_column_number</i> | 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

|                      |  |
|----------------------|--|
| <i>column_number</i> | the number of the column to look for matches |
| <i>value</i>         | 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

|                      |                               |
|----------------------|-------------------------------|
| <i>row_number</i>    | the row number of the cell    |
| <i>column_number</i> | 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

|                    |                        |
|--------------------|------------------------|
| <i>column_name</i> | 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**

|                  |                                 |
|------------------|---------------------------------|
| <i>key_value</i> | 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**

|               |                             |
|---------------|-----------------------------|
| <i>values</i> | 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**

|                   |  |
|-------------------|--|
| <i>row_number</i> | the number of the row to replace       |
| <i>values</i>     | 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

|                     |   |
|---------------------|---|
| <i>target_frame</i> | 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**

|              |  |
|--------------|--|
| <i>depth</i> | 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

|                     |   |
|---------------------|---|
| <i>target_frame</i> | 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

|              |  |
|--------------|--|
| <i>depth</i> | 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

|                       |                                 |
|-----------------------|---------------------------------|
| <i>ssid</i>           | The SSID of the analysed frames |
| <i>code_file_name</i> | 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

|               |   |
|---------------|---|
| <i>values</i> | 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

|                    |                             |
|--------------------|-----------------------------|
| <i>key_value</i>   | the value of the key        |
| <i>column_name</i> | 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

|                    |                                   |
|--------------------|-----------------------------------|
| <i>column_name</i> | the name of the column to compare |
| <i>cell_value</i>  | 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

|                  |  |
|------------------|--|
| <i>key_value</i> | the key value of the entry to replace                  |
| <i>values</i>    | 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

|                            |                                      |
|----------------------------|--------------------------------------|
| <code>_communicator</code> | 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**

|              |   |
|--------------|---|
| <i>field</i> | 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

|            |                 |
|------------|-----------------|
| <i>arg</i> | 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

|              |  |
|--------------|--|
| <i>depth</i> | 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

|                   |  |
|-------------------|--|
| <i>field_name</i> | 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

|            |                 |
|------------|-----------------|
| <i>arg</i> | 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

|                     |   |
|---------------------|---|
| <i>target_frame</i> | 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

|              |  |
|--------------|--|
| <i>depth</i> | 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

- [le\\_node](#) (uint8\_t element\_id, [Big\\_number](#) value)  
*Construct a new [le\\_node](#) object.*
- [~le\\_node](#) ()  
*Destroy the [le\\_node](#) object.*
- void [add](#) ([le\\_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 [le\\_node](#)()

```
Ie_node::Ie_node (
    uint8_t element_id,
    Big\_number value )
```

Construct a new [le\\_node](#) object.

##### Parameters

|                   |                              |
|-------------------|------------------------------|
| <i>element_id</i> | the element id of the new IE |
| <i>value</i>      | the value of the new IE      |

#### 6.13.2.2 [~le\\_node](#)()

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

Destroy the [le\\_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

|                    |                     |
|--------------------|---------------------|
| <i>new_element</i> | 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

|              |   |
|--------------|---|
| <i>field</i> | the name of the field or its element id |
|--------------|---|

#### Returns

[Big\\_number](#) the value that correspond 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

|            |                 |
|------------|-----------------|
| <i>arg</i> | 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

|                     |   |
|---------------------|---|
| <i>target_frame</i> | 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

|              |  |
|--------------|--|
| <i>depth</i> | 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

|            |                 |
|------------|-----------------|
| <i>arg</i> | 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

|                     |   |
|---------------------|---|
| <i>target_frame</i> | 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**

|              |  |
|--------------|--|
| <i>depth</i> | 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

|                     |   |
|---------------------|---|
| <i>target_frame</i> | 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

|              |  |
|--------------|--|
| <i>depth</i> | 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

|              |   |
|--------------|---|
| <i>value</i> | the fixed value of the <a href="#">Node</a> |
|--------------|---|



## 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

|            |                 |
|------------|-----------------|
| <i>arg</i> | 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

|                     |   |
|---------------------|---|
| <i>target_frame</i> | 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

|              |  |
|--------------|--|
| <i>depth</i> | 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](#)  
*Transform 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 {
10     class Compiler {
11     private:
12         const os_communicator::Communicator *_communicator;
13
14         // return the next line to read and fill the current_node
15         size_t read_line(executable_tree::Node *current_node, size_t next_line) const;
16
17     public:
18         Compiler(const os_communicator::Communicator *communicator)
19             : _communicator(communicator) {
20             if(!_communicator)
21                 throw invalid_argument("No communicator given");
22         };
23
24         executable_tree::Node *get_executable_tree() const;
25     };
26 }
27
28 #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{
21     class Sha256 : public Function {
22     public:
23         std::string to_string(size_t depth) const override;
24
25         std::string get_value(const decoder::Frame *target_frame) const override;
26     } ;
27
28     class Cut_bit : public Function {
29     public:
30         std::string to_string(size_t depth) const override;
31
32         std::string get_value(const decoder::Frame *target_frame) const override;
33     } ;
34
35     class Cut_byte : public Function {
36     public:
37         std::string to_string(size_t depth) const override;
38
39         std::string get_value(const decoder::Frame *target_frame) const override;
40     } ;
41 }
42
43 #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

Copyright (c) 2025

## 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{
20     class Node {
21     public:
22         virtual std::string get_value(const decoder::Frame *target_frame) const;
23
24         virtual void add_node(Node* arg);
25
26         virtual std::string to_string(size_t depth) const;
27     };
28
29     class Root : public Node {
30     private:
31         Node* _first_node = nullptr;
32     public:
33         ~Root(){
34             if(_first_node)
35                 delete _first_node;
36         };
37
38         std::string get_value(const decoder::Frame *target_frame) const override;
39
40         void add_node(Node* arg) override;
41
42         std::string to_string(size_t depth) const override;
43     };
44
45     class Function : public Node {
46     protected:
47         std::vector<Node*> args;
48     public:
49         ~Function(){
50             for (Node* arg : args)
51                 delete arg;
52         }
53     };
54 }
```

```

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
103 class Value : public Node {
104 private:
105     std::string value;
106
107 public:
108     Value(std::string value) : value(value) {};
109
110     std::string get_value(const decoder::Frame *target_frame) const override;
111
112     void add_node(Node* arg) override;
113
114     std::string to_string(size_t depth) const override;
115 };
116
117 class Getter : public Node {
118 private:
119     const std::string field_name;
120
121 public:
122     Getter(const std::string field_name)
123         : field_name(field_name) {};
124
125     std::string get_value(const decoder::Frame *target_frame) const override;
126
127     void add_node(Node* arg) override;
128
129     std::string to_string(size_t depth) const override;
130 };
131 }
132 #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

Copyright (c) 2025

### 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

Copyright (c) 2025

### 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{
34     class Csv{
35     private:
36         size_t _number_of_columns;
37         size_t _key_column_number;
38         os_communicator::Communicator *_c_metadata;
39         os_communicator::Communicator *_c_data;
40
41         void _test_values_size(std::vector<std::string> values);
42         size_t _get_key_position(std::string key_value);
43         static std::string _values_to_line(std::vector<std::string> values);
44         size_t _get_column_position_from_header(std::string header, std::string column_name);
45         static std::string _get_cell_from_row(std::string row, size_t column_number);
46         size_t _string_to_size_t(std::string number);
47
48     public:
49         Csv(std::string file_name);
50         ~Csv();
51         static Csv *create(std::string file_name, std::vector<std::string> column_names, size_t
52 key_column_number);
53
54         void push_row(std::vector<std::string> values);
55         void replace_row(size_t row_number, std::vector<std::string> values);
56         std::string get_cell(size_t row_number, size_t column_number);
57         size_t get_row_number(std::string key_value);
58         size_t get_column_number(std::string column_name);
59         std::vector<std::string> get_all_keys();
60         std::vector<std::string> get_all_keys_with_value(size_t column_number, std::string value);
61     };
62
63     class Database{
64     private:
65         std::string _ssid;
66         std::string _code_file_name;
67         std::string _code_hash;
68         Csv *_csv;
69
70     public:
71         Database(std::string ssid, std::string code_file_name);
72         ~Database();
73
74         std::string get_cell(std::string key_value, std::string column_name);
75         std::vector<std::string> get_key_of_entries(std::string column_name, std::string
76 cell_value);
77         void add_entry(std::vector<std::string> values);
78         void replace_entry(std::string key_value, std::vector<std::string> values);
79     };
80 };
81
82 #endif

```

## 7.11 includes/decoder/big\_number.hpp File Reference

```

#include <vector>
#include <cstdint>

```

```
#include <cstdint>
#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 <cstdint>
17 #include <stdexcept>
18 #include <string>
19
20
21 namespace decoder{
22     class Big_number {
23     private:
24         std::vector<uint8_t> number;
25         bool _is_null = false;
26
27         Big_number& operator=(const std::vector<uint8_t> _number);
28
29     public:
30         bool is_null() const;
31         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

Copyright (c) 2025

## 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{
25     class Body {
26     protected:
27         uint8_t *_raw_body_buffer;
28         size_t _raw_buffer_size;
29
30         virtual void decode() = 0;
31
32     public:
33         Body(uint8_t *raw_body_buffer, size_t raw_buffer_size);
34
35         virtual void print() const = 0;
36         virtual Big_number get_value(string field) const = 0;
37
38         static Body *get_body(uint8_t *raw_body_buffer, size_t raw_buffer_size, size_t type, size_t
sub_type);
39     };
40
41     class Frame{
42     private:
43         os_communicator::Communicator *communicator;
44     protected:
45         uint8_t *raw_frame_buffer;
46         size_t raw_buffer_size;
47         size_t raw_frame_size;
48         bool is_decoded;
49         bool has_raw_data;
50
51         // Header
52         Big_number frame_control = Big_number::null();
53         Big_number duration = Big_number::null();
54         Big_number destination_address = Big_number::null();
55         Big_number source_address = Big_number::null();
56         Big_number bssid = Big_number::null();
57         Big_number sequence_control = Big_number::null();
58         Big_number frame_check_sum = Big_number::null();
59
60         // Body
61         Body *body = nullptr;
62
63     public:
64         Frame(os_communicator::Communicator *_communicator);
65         ~Frame();
66
67         bool get_is_decoded() const;
68         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 <stdint>
#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

Copyright (c) 2025

### 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 rates

#### 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 <stdint>
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{
30     class Ie_node {
31     private:
32         uint8_t element_id;
33         Big_number element_value;
34         Ie_node *next_element;
35
36         bool is_valid_hex(const std::string str) const;
37
38     public:
39         Ie_node(uint8_t element_id, Big_number value);
40         ~Ie_node();
41
42         void add(Ie_node *new_element);
43         Big_number get_value(std::string field) const;
44         void print();
45     };
46 }
47
48 #endif

```

## 7.17 includes/decoder/management\_body.hpp File Reference

```

#include <stdint>
#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

Copyright (c) 2025

## 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{
24     class Beacon_body : public Body {
25     private:
26         Big_number _timestamp;
27         Big_number _beacon_interval;
28         Big_number _capabilities_information;
29         Ie_node *_first_ie = nullptr;
30
31         void decode() override;
32         void add_ie(uint8_t element_id, uint8_t element_length, size_t start_position);
33
34     public:
35         Beacon_body(uint8_t *raw_body_buffer, size_t raw_buffer_size);
36         ~Beacon_body();
37
38         void print() const override;
39         Big_number get_value(string field) const override;
40     };
41 }
42
43 #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 <cstdint>
#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

Copyright (c) 2025

## 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 <cstdint>
16 #include <cstdlib>
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
34     class Communicator{
35     private:
36         string _file_name;
37
38     public:
39         Communicator(string file_name);
40
41         bool exist();
42         size_t in_buffer(uint8_t *buffer, const size_t buffer_size);
43     }
```

```
72         string get_line(size_t line_number) const;
73         void add_line(string line);
85         void replace_line(size_t line_number, string new_line);
90         void new_file();
91
92         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

- `_raw_body_buffer`
    - `decoder::Body`, [20](#)
  - `_raw_buffer_size`
    - `decoder::Body`, [20](#)
  - `~Beacon_body`
    - `decoder::Beacon_body`, [12](#)
  - `~Csv`
    - `database::Csv`, [28](#)
  - `~Database`
    - `database::Database`, [35](#)
  - `~Frame`
    - `decoder::Frame`, [39](#)
  - `~Function`
    - `executable_tree::Function`, [43](#)
  - `~le_node`
    - `decoder::le_node`, [47](#)
  - `~Root`
    - `executable_tree::Root`, [51](#)
- `add`
  - `decoder::le_node`, [47](#)
- `add_entry`
  - `database::Database`, [35](#)
- `add_line`
  - `os_communicator::Communicator`, [21](#)
- `add_node`
  - `executable_tree::Function`, [43](#)
  - `executable_tree::Getter`, [45](#)
  - `executable_tree::Node`, [49](#)
  - `executable_tree::Root`, [51](#)
  - `executable_tree::Value`, [55](#)
- `args`
  - `executable_tree::Function`, [44](#)
- `Beacon_body`
  - `decoder::Beacon_body`, [11](#)
- `BEACON_FRAME_BODY_MIN_LENGTH`
  - `const.hpp`, [62](#)
- `BEACON_FRAME_MAX_LENGTH`
  - `const.hpp`, [62](#)
- `Body`
  - `decoder::Body`, [18](#)
- `body`
  - `decoder::Frame`, [40](#)
- `bssid`
  - `decoder::Frame`, [40](#)
- `char_string`
  - `decoder::Big_number`, [13](#)
- `CHARACTER_DEVICE_FILE`
  - `main.cpp`, [79](#)
- `COLUMN_NAMES`
  - `core.hpp`, [64](#)
- `COLUMN_NUMBER_LINE_NUMBER`
  - `core.hpp`, [65](#)
- `Communicator`
  - `os_communicator::Communicator`, [21](#)
- `Compiler`
  - `compiler::Compiler`, [25](#)
- `compiler`, [9](#)
- `compiler::Compiler`, [25](#)
  - `Compiler`, [25](#)
  - `get_executable_tree`, [25](#)
- `const.hpp`
  - `BEACON_FRAME_BODY_MIN_LENGTH`, [62](#)
  - `BEACON_FRAME_MAX_LENGTH`, [62](#)
  - `DATABASE_ROOT`, [62](#)
  - `FRAME_HEADER_MAX_LENGTH`, [62](#)
  - `FRAME_HEADER_MIN_LENGTH`, [63](#)
  - `FRAME_MAX_LENGTH`, [63](#)
- `copy`
  - `decoder::Big_number`, [14](#)
- `core.hpp`
  - `COLUMN_NAMES`, [64](#)
  - `COLUMN_NUMBER_LINE_NUMBER`, [65](#)
  - `DATA_EXTENTION`, [65](#)
  - `DELIMITER`, [65](#)
  - `HEADER_SIZE`, [65](#)
  - `KEY_COLUMN_NUMBER`, [65](#)
  - `KEY_LINE_NUMBER`, [65](#)
  - `METADATA_EXTENTION`, [65](#)
- `create`
  - `database::Csv`, [28](#)
- `create_folder`
  - `os_communicator::Communicator`, [22](#)
- `Csv`
  - `database::Csv`, [27](#)
- `cut_bit`
  - `decoder::Big_number`, [14](#)
- `cut_byte`
  - `decoder::Big_number`, [14](#)
- `DATA_EXTENTION`
  - `core.hpp`, [65](#)
- `Database`
  - `database::Database`, [34](#)
- `database`, [9](#)
  - `hash`, [9](#)
- `database::Csv`, [26](#)
  - `~Csv`, [28](#)

- create, 28
- Csv, 27
- get\_all\_keys, 28
- get\_all\_keys\_with\_value, 28
- get\_cell, 29
- get\_column\_number, 29
- get\_row\_number, 30
- push\_row, 30
- replace\_row, 30
- database::Database, 34
  - ~Database, 35
  - add\_entry, 35
  - Database, 34
  - get\_cell, 35
  - get\_key\_of\_entries, 35
  - replace\_entry, 36
- DATABASE\_ROOT
  - const.hpp, 62
- decode
  - decoder::Body, 18
  - decoder::Frame, 39
- decoder, 10
- decoder::Beacon\_body, 11
  - ~Beacon\_body, 12
  - Beacon\_body, 11
  - get\_value, 12
  - print, 12
- decoder::Big\_number, 13
  - char\_string, 13
  - copy, 14
  - cut\_bit, 14
  - cut\_byte, 14
  - from\_buffer, 15
  - from\_buffer\_inv, 15
  - from\_hex\_string, 15
  - hex\_string, 16
  - is\_null, 16
  - null, 16
  - operator=, 16
  - throw\_if\_null, 17
  - to\_size\_t, 17
- decoder::Body, 17
  - \_raw\_body\_buffer, 20
  - \_raw\_buffer\_size, 20
  - Body, 18
  - decode, 18
  - get\_body, 19
  - get\_value, 19
  - print, 19
- decoder::Frame, 36
  - ~Frame, 39
  - body, 40
  - bssid, 40
  - decode, 39
  - destination\_address, 41
  - duration, 41
  - Frame, 37
  - frame\_check\_sum, 41
  - frame\_control, 41
  - get\_is\_decoded, 39
  - get\_value, 39
  - has\_raw\_data, 41
  - is\_decoded, 41
  - print, 40
  - print\_raw\_data, 40
  - raw\_buffer\_size, 41
  - raw\_frame\_buffer, 42
  - raw\_frame\_size, 42
  - sequence\_control, 42
  - source\_address, 42
  - update\_raw\_data, 40
- decoder::le\_node, 46
  - ~le\_node, 47
  - add, 47
  - get\_value, 48
  - le\_node, 47
  - print, 48
- DELIMITER
  - core.hpp, 65
- destination\_address
  - decoder::Frame, 41
- duration
  - decoder::Frame, 41
- executable\_tree, 10
- executable\_tree::Cut\_bit, 31
  - get\_value, 31
  - to\_string, 32
- executable\_tree::Cut\_byte, 32
  - get\_value, 33
  - to\_string, 33
- executable\_tree::Function, 42
  - ~Function, 43
  - add\_node, 43
  - args, 44
  - to\_string, 44
- executable\_tree::Getter, 44
  - add\_node, 45
  - get\_value, 46
  - Getter, 45
  - to\_string, 46
- executable\_tree::Node, 48
  - add\_node, 49
  - get\_value, 49
  - to\_string, 50
- executable\_tree::Root, 50
  - ~Root, 51
  - add\_node, 51
  - get\_value, 51
  - to\_string, 52
- executable\_tree::Sha256, 52
  - get\_value, 53
  - to\_string, 53
- executable\_tree::Value, 54
  - add\_node, 55
  - get\_value, 55
  - to\_string, 55

- Value, 54
- exist
  - os\_communicator::Communicator, 22
- F\_FILE
  - os\_communicator.hpp, 76
- F\_FOLDER
  - os\_communicator.hpp, 76
- F\_NONE
  - os\_communicator.hpp, 76
- Frame
  - decoder::Frame, 37
- frame\_check\_sum
  - decoder::Frame, 41
- frame\_control
  - decoder::Frame, 41
- FRAME\_HEADER\_MAX\_LENGTH
  - const.hpp, 62
- FRAME\_HEADER\_MIN\_LENGTH
  - const.hpp, 63
- FRAME\_MAX\_LENGTH
  - const.hpp, 63
- from\_buffer
  - decoder::Big\_number, 15
- from\_buffer\_inv
  - decoder::Big\_number, 15
- from\_hex\_string
  - decoder::Big\_number, 15
- get\_all\_keys
  - database::Csv, 28
- get\_all\_keys\_with\_value
  - database::Csv, 28
- get\_body
  - decoder::Body, 19
- get\_cell
  - database::Csv, 29
  - database::Database, 35
- get\_column\_number
  - database::Csv, 29
- get\_current\_date
  - os\_communicator::Communicator, 22
- get\_executable\_tree
  - compiler::Compiler, 25
- get\_file\_type
  - os\_communicator::Communicator, 22
- get\_is\_decoded
  - decoder::Frame, 39
- get\_key\_of\_entries
  - database::Database, 35
- get\_line
  - os\_communicator::Communicator, 23
- get\_row\_number
  - database::Csv, 30
- get\_value
  - decoder::Beacon\_body, 12
  - decoder::Body, 19
  - decoder::Frame, 39
  - decoder::le\_node, 48
  - executable\_tree::Cut\_bit, 31
  - executable\_tree::Cut\_byte, 33
  - executable\_tree::Getter, 46
  - executable\_tree::Node, 49
  - executable\_tree::Root, 51
  - executable\_tree::Sha256, 53
  - executable\_tree::Value, 55
- Getter
  - executable\_tree::Getter, 45
- has\_raw\_data
  - decoder::Frame, 41
- hash
  - database, 9
- HEADER\_SIZE
  - core.hpp, 65
- hex\_string
  - decoder::Big\_number, 16
- ie.hpp
  - P\_CF, 71
  - P\_Challenge\_text, 71
  - P\_DS, 71
  - P\_FH, 71
  - P\_IBSS, 72
  - P\_SSID, 72
  - P\_SUPPORTED\_RATES, 72
  - P\_TIM, 72
- le\_node
  - decoder::le\_node, 47
- in\_buffer
  - os\_communicator::Communicator, 23
- includes/compiler/compiler.hpp, 57
- includes/compiler/function\_node.hpp, 58, 59
- includes/compiler/node.hpp, 59, 60
- includes/const.hpp, 61, 63
- includes/database/core.hpp, 63, 66
- includes/decoder/big\_number.hpp, 66, 67
- includes/decoder/frame.hpp, 68, 69
- includes/decoder/ie.hpp, 70, 73
- includes/decoder/management\_body.hpp, 73, 74
- includes/os\_communicator/os\_communicator.hpp, 75, 76
- is\_decoded
  - decoder::Frame, 41
- is\_null
  - decoder::Big\_number, 16
- KEY\_COLUMN\_NUMBER
  - core.hpp, 65
- KEY\_LINE\_NUMBER
  - core.hpp, 65
- main
  - main.cpp, 80
- main.cpp
  - CHARACTER\_DEVICE\_FILE, 79
  - main, 80
  - PERIOD, 79

- run, 80
- SCRIPT\_FILE, 79
- METADATA\_EXTENTION
  - core.hpp, 65
- new\_file
  - os\_communicator::Communicator, 23
- notify
  - os\_communicator::Communicator, 24
- null
  - decoder::Big\_number, 16
- operator=
  - decoder::Big\_number, 16
- os\_communicator, 10
- os\_communicator.hpp
  - F\_FILE, 76
  - F\_FOLDER, 76
  - F\_NONE, 76
- os\_communicator::Communicator, 20
  - add\_line, 21
  - Communicator, 21
  - create\_folder, 22
  - exist, 22
  - get\_current\_date, 22
  - get\_file\_type, 22
  - get\_line, 23
  - in\_buffer, 23
  - new\_file, 23
  - notify, 24
  - replace\_line, 24
  - sleep, 24
- P\_CF
  - ie.hpp, 71
- P\_Challenge\_text
  - ie.hpp, 71
- P\_DS
  - ie.hpp, 71
- P\_FH
  - ie.hpp, 71
- P\_IBSS
  - ie.hpp, 72
- P\_SSID
  - ie.hpp, 72
- P\_SUPPORTED\_RATES
  - ie.hpp, 72
- P\_TIM
  - ie.hpp, 72
- PERIOD
  - main.cpp, 79
- print
  - decoder::Beacon\_body, 12
  - decoder::Body, 19
  - decoder::Frame, 40
  - decoder::le\_node, 48
- print\_raw\_data
  - decoder::Frame, 40
- push\_row
  - database::Csv, 30
- raw\_buffer\_size
  - decoder::Frame, 41
- raw\_frame\_buffer
  - decoder::Frame, 42
- raw\_frame\_size
  - decoder::Frame, 42
- replace\_entry
  - database::Database, 36
- replace\_line
  - os\_communicator::Communicator, 24
- replace\_row
  - database::Csv, 30
- run
  - main.cpp, 80
- SCRIPT\_FILE
  - main.cpp, 79
- sequence\_control
  - decoder::Frame, 42
- sleep
  - os\_communicator::Communicator, 24
- source\_address
  - decoder::Frame, 42
- src/compiler/compiler.cpp, 77
- src/compiler/function\_node.cpp, 77
- src/compiler/node.cpp, 77
- src/database/csv.cpp, 77
- src/database/database.cpp, 77
- src/decoder/big\_number.cpp, 78
- src/decoder/frame.cpp, 78
- src/decoder/ie.cpp, 78
- src/decoder/management\_body.cpp, 78
- src/main.cpp, 78
- src/os\_communicator/os\_communicator.cpp, 80
- throw\_if\_null
  - decoder::Big\_number, 17
- to\_size\_t
  - decoder::Big\_number, 17
- to\_string
  - executable\_tree::Cut\_bit, 32
  - executable\_tree::Cut\_byte, 33
  - executable\_tree::Function, 44
  - executable\_tree::Getter, 46
  - executable\_tree::Node, 50
  - executable\_tree::Root, 52
  - executable\_tree::Sha256, 53
  - executable\_tree::Value, 55
- update\_raw\_data
  - decoder::Frame, 40
- Value
  - executable\_tree::Value, 54