

Reference Manual

Generated by Doxygen 1.7.4

Tue Oct 11 2011 11:53:09

Contents

1	Engine queries	1
2	Scan control functions	1
3	Bytecode configuration	2
4	String operations	3
5	PDF handling functions	4
6	Environment detection functions	5
7	PE functions	5
8	JS normalize API	9
9	Icon matcher APIs	9
10	Math functions	9
11	Data structure handling functions	9
12	File operations	12
13	Global variables	12
14	Disassemble APIs	13
15	Data Structure Documentation	13
15.1	cli_exe_info Struct Reference	13
15.1.1	Detailed Description	13
15.1.2	Field Documentation	13
15.2	cli_exe_section Struct Reference	14
15.2.1	Detailed Description	14
15.2.2	Field Documentation	14
15.3	cli_pe_hook_data Struct Reference	15
15.3.1	Detailed Description	15
15.3.2	Field Documentation	16

15.4 DIS_arg Struct Reference	16
15.4.1 Detailed Description	17
15.4.2 Field Documentation	17
15.5 DIS_fixed Struct Reference	17
15.5.1 Detailed Description	17
15.5.2 Field Documentation	18
15.6 DIS_mem_arg Struct Reference	18
15.6.1 Detailed Description	18
15.6.2 Field Documentation	18
15.7 DISASM_RESULT Struct Reference	19
15.7.1 Detailed Description	19
15.8 pe_image_data_dir Struct Reference	19
15.8.1 Detailed Description	19
15.9 pe_image_file_hdr Struct Reference	19
15.9.1 Detailed Description	19
15.9.2 Field Documentation	20
15.10 pe_image_optional_hdr32 Struct Reference	20
15.10.1 Detailed Description	21
15.10.2 Field Documentation	21
15.11 pe_image_optional_hdr64 Struct Reference	22
15.11.1 Detailed Description	22
15.11.2 Field Documentation	22
15.12 pe_image_section_hdr Struct Reference	24
15.12.1 Detailed Description	24
15.12.2 Field Documentation	24
16 File Documentation	25
16.1 bytecode_api.h File Reference	25
16.1.1 Detailed Description	27
16.1.2 Enumeration Type Documentation	27
16.1.3 Function Documentation	28
16.1.4 Variable Documentation	53
16.2 bytecode_disasm.h File Reference	54
16.2.1 Detailed Description	56

16.2.2 Enumeration Type Documentation	56
16.3 bytecode_execs.h File Reference	64
16.3.1 Detailed Description	65
16.4 bytecode_local.h File Reference	65
16.4.1 Detailed Description	67
16.4.2 Define Documentation	67
16.4.3 Function Documentation	71
16.5 bytecode_pe.h File Reference	90
16.5.1 Detailed Description	90

1 Engine queries

Global [count_match](#)(__Signature sig)

Global [engine_db_options](#)(void)

Global [engine_dconf_level](#)(void)

Global [engine_functionality_level](#)(void)

Global [engine_scan_options](#)(void)

Global [match_location](#)(__Signature sig, uint32_t goback)

Global [match_location_check](#)(__Signature sig, uint32_t goback, const char *static_start, uint32_t static_len)
It is recommended to use this for safety and compatibility with 0.96.1

Global [matches](#)(__Signature sig)

2 Scan control functions

Global [bytecode_rt_error](#)(int32_t locationid)

Global [extract_new](#)(int32_t id)

Global [extract_set_container](#)(uint32_t container)

Global [foundVirus](#)(const char *virusname)

Global [input_switch](#)(int32_t extracted_file)

Global [setvirusname](#)(const uint8_t *name, uint32_t len)

3 Bytecode configuration

Global [COPYRIGHT\(c\)](#) This will also prevent the sourcecode from being embedded into the bytecode

Global [DECLARE_SIGNATURE](#)(name)

Global [DEFINE_SIGNATURE](#)(name, hex)

Global [FUNCTIONALITY_LEVEL_MAX](#)(m)

Global [FUNCTIONALITY_LEVEL_MIN](#)(m)

Global [ICONGROUP1](#)(group)

Global [ICONGROUP2](#)(group)

Global [PDF_HOOK_DECLARE](#) This hook is called several times, use [pdf_get_phase\(\)](#) to find out in which phase you got called.

Global [PE_HOOK_DECLARE](#)

Global [PE_UNPACKER_DECLARE](#)

Global [SIGNATURES_DECL_BEGIN](#)

Global [SIGNATURES_DECL_END](#)

Global [SIGNATURES_DEF_BEGIN](#)

Global [SIGNATURES_END](#)

Global [TARGET](#)(tgt)

Global [VIRUSNAME_PREFIX](#)(name)

Global [VIRUSNAMES](#)(...)

4 String operations

Global [atoi](#)(const uint8_t *str, int32_t size)

Global [debug_print_str](#)(const uint8_t *str, uint32_t len)

Global [debug_print_str_nonl](#)(const uint8_t *str, uint32_t len)

Global [debug_print_str_start](#)(const uint8_t *str, uint32_t len)

Global [debug_print_uint](#)(uint32_t a)

Global [entropy_buffer](#)(uint8_t *buffer, int32_t size)

Global [hex2ui](#)(uint32_t hex1, uint32_t hex2)

Global [memchr](#)(const void *s, int c, size_t n)

Global [memcmp](#)(const void *s1, const void *s2, uint32_t n) [__attribute__\(\(__nothrow__\)\)](#) [__attribute__\(\(__pure__\)\)](#)

Global `memcpy`(void *restrict dst, const void *restrict src, uintptr_t n) __attribute__((__nothrow__)) __attribute__((__nonnull__((1))))

Global `memmove`(void *dst, const void *src, uintptr_t n) __attribute__((__nothrow__)) __attribute__((__nonnull__((1))))

Global `memset`(void *src, int c, uintptr_t n) __attribute__((__nothrow__)) __attribute__((__nonnull__((1))))

Global `memstr`(const uint8_t *haystack, int32_t haysize, const uint8_t *needle, int32_t needlesize)

5 PDF handling functions

Global `pdf_get_dumpedobjid`(void) Valid only in PDF_PHASE_POSTDUMP.

Global `pdf_get_flags`(void)

Global `pdf_get_obj_num`(void)

Global `pdf_get_phase`(void) Identifies at which phase this bytecode was called.

Global `pdf_getobj`(int32_t objidx, uint32_t amount) Meant only for reading, write modifies the fmap buffer, so avoid!

Global `pdf_getobjsize`(int32_t objidx)

Global `pdf_lookupobj`(uint32_t id)

Global `pdf_set_flags`(int32_t flags)

6 Environment detection functions

Global `__is_bigendian`(void) `__attribute__((const)) __attribute__((nothrow))`

Global `check_platform`(uint32_t a, uint32_t b, uint32_t c)

Global `disable_bytecode_if`(const int8_t *reason, uint32_t len, uint32_t cond)

Global `disable_jit_if`(const int8_t *reason, uint32_t len, uint32_t cond)

Global `get_environment`(struct cli_environment *env, uint32_t len)

Global `version_compare`(const uint8_t *lhs, uint32_t lhs_len, const uint8_t *rhs, uint32_t rhs_len)

7 PE functions

Class `cli_exe_info`

Class `cli_exe_section`

Class `cli_pe_hook_data`

Global `get_pe_section`(struct `cli_exe_section` *section, uint32_t num)

Global `getEntryPoint`(void)

Global `getExeOffset`(void)

Global `getImageBase`(void)

Global `getNumberOfSections`(void)

Global `getPEBaseOfCode`(void)

Global [getPEBaseOfData\(void\)](#)

Global [getPECharacteristics\(\)](#)

Global [getPEChecksum\(void\)](#)

Global [getPEDataDirRVA\(unsigned n\)](#)

Global [getPEDataDirSize\(unsigned n\)](#)

Global [getPEDllCharacteristics\(void\)](#)

Global [getPEFileAlignment\(void\)](#)

Global [getPEImageBase\(void\)](#)

Global [getPEisDLL\(\)](#)

Global [getPELFANew\(void\)](#)

Global [getPELoaderFlags\(void\)](#)

Global [getPEMachine\(\)](#)

Global [getPEMajorImageVersion\(void\)](#)

Global [getPEMajorLinkerVersion\(void\)](#)

Global [getPEMajorOperatingSystemVersion\(void\)](#)

Global [getPEMajorSubsystemVersion\(void\)](#)

Global [getPEMinorImageVersion\(void\)](#)

Global [getPEMinorLinkerVersion\(void\)](#)

Global [getPEMinorOperatingSystemVersion\(void\)](#)

Global [getPEMinorSubsystemVersion\(void\)](#)

Global [getPENumberOfSymbols\(\)](#)

Global [getPEPointerToSymbolTable\(\)](#)

Global [getPESectionAlignment\(void\)](#)

Global [getPESizeOfCode\(void\)](#)

Global [getPESizeOfHeaders\(void\)](#)

Global [getPESizeOfHeapCommit\(void\)](#)

Global [getPESizeOfHeapReserve\(void\)](#)

Global [getPESizeOfImage\(void\)](#)

Global [getPESizeOfInitializedData\(void\)](#)

Global [getPESizeOfOptionalHeader\(\)](#)

Global [getPESizeOfStackCommit\(void\)](#)

Global [getPESizeOfStackReserve\(void\)](#)

Global [getPESizeOfUninitializedData\(void\)](#)

Global [getPESubsystem\(void\)](#)

Global [getPETimeStamp\(\)](#)

Global [getPEWin32VersionValue\(void\)](#)

Global [getSectionRVA\(unsigned i\)](#) .

Global [getSectionVirtualSize\(unsigned i\)](#) .

Global [getVirtualEntryPoint\(void\)](#)

Global [hasExeInfo\(void\)](#)

Global [hasPEInfo\(void\)](#)

Global [isPE64\(void\)](#)

Class [pe_image_data_dir](#)

Class [pe_image_file_hdr](#)

Class [pe_image_optional_hdr32](#)

Class [pe_image_optional_hdr64](#)

Class [pe_image_section_hdr](#)

Global [pe_rawaddr\(uint32_t rva\)](#)

Global [readPESectionName\(unsigned char name\[8\], unsigned n\)](#)

Global [readRVA\(uint32_t rva, void *buf, size_t bufsize\)](#)

8 JS normalize API

Global `jsnorm_done`(int32_t id)

Global `jsnorm_init`(int32_t from_buffer)

Global `jsnorm_process`(int32_t id)

9 Icon matcher APIs

Global `matchicon`(const uint8_t *group1, int32_t group1_len, const uint8_t *group2, int32_t group2_len)

10 Math functions

Global `icos`(int32_t a, int32_t b, int32_t c)

Global `ieexp`(int32_t a, int32_t b, int32_t c)

Global `ilog2`(uint32_t a, uint32_t b)

Global `ipow`(int32_t a, int32_t b, int32_t c)

Global `isin`(int32_t a, int32_t b, int32_t c)

11 Data structure handling functions

Global `buffer_pipe_done`(int32_t id) After this all attempts to use this buffer will result in error. All buffer_pipes are automatically deallocated when bytecode finishes execution.

Global `buffer_pipe_new`(uint32_t size)

Global `buffer_pipe_new_fromfile`(uint32_t pos)

Global **buffer_pipe_read_avail**(int32_t id)

Global **buffer_pipe_read_get**(int32_t id, uint32_t amount) The 'amount' parameter should be obtained by a call to **buffer_pipe_read_avail**().

Global **buffer_pipe_read_stopped**(int32_t id, uint32_t amount) Updates read cursor in buffer_pipe.

Global **buffer_pipe_write_avail**(int32_t id)

Global **buffer_pipe_write_get**(int32_t id, uint32_t size) Returns pointer to writable buffer. The 'amount' parameter should be obtained by a call to **buffer_pipe_write_avail**().

Global **buffer_pipe_write_stopped**(int32_t id, uint32_t amount)

Global **cli_readint16**(const void *buff)

Global **cli_readint32**(const void *buff)

Global **cli_writeint32**(void *offset, uint32_t v)

Global **hashset_add**(int32_t hs, uint32_t key)

Global **hashset_contains**(int32_t hs, uint32_t key)

Global **hashset_done**(int32_t id) Trying to use the hashset after this will result in an error. The hashset may not be used after this. All hashsets are automatically deallocated when bytecode finishes execution.

Global **hashset_empty**(int32_t id)

Global **hashset_new**(void)

Global **hashset_remove**(int32_t hs, uint32_t key)

Global `inflate_done`(int32_t id)

Global `inflate_init`(int32_t from_buffer, int32_t to_buffer, int32_t windowBits) 'from_buffer' and writing uncompressed data 'to_buffer'.

Global `inflate_process`(int32_t id)

Global `le16_to_host`(uint16_t v)

Global `le32_to_host`(uint32_t v)

Global `le64_to_host`(uint64_t v)

Global `malloc`(uint32_t size)

Global `map_addkey`(const uint8_t *key, int32_t ksize, int32_t id)

Global `map_done`(int32_t id)

Global `map_find`(const uint8_t *key, int32_t ksize, int32_t id)

Global `map_getvalue`(int32_t id, int32_t size)

Global `map_getvaluesize`(int32_t id)

Global `map_new`(int32_t keysize, int32_t valuesize)

Global `map_remove`(const uint8_t *key, int32_t ksize, int32_t id)

Global `map_setvalue`(const uint8_t *value, int32_t vsize, int32_t id)

12 File operations

Global `buffer_pipe_new_fromfile(uint32_t pos)` to the current file, at the specified position.

Global `file_byteat(uint32_t offset)`

Global `file_find(const uint8_t *data, uint32_t len)`

Global `file_find_limit(const uint8_t *data, uint32_t len, int32_t maxpos)`

Global `fill_buffer(uint8_t *buffer, uint32_t len, uint32_t filled, uint32_t cursor, uint32_t fill)`

Global `getFileSize(void)`

Global `read(uint8_t *data, int32_t size)`

Global `read_number(uint32_t radix)` Non-numeric characters are ignored.

Global `seek(int32_t pos, uint32_t whence)`

Global `write(uint8_t *data, int32_t size)`

13 Global variables

Global `__clambc_filesize[1]`

Global `__clambc_kind`

Global `__clambc_match_counts[64]`

Global `__clambc_match_offsets[64]`

Global `__clambc_pedata`

14 Disassemble APIs

Class [DIS_arg](#)

Class [DIS_fixed](#)

Class [DIS_mem_arg](#)

Global [disasm_x86](#)(struct [DISASM_RESULT](#) *result, uint32_t len)

Global [DisassembleAt](#)(struct [DIS_fixed](#) *result, uint32_t offset, uint32_t len)

15 Data Structure Documentation

15.1 cli_exe_info Struct Reference

Data Fields

- struct [cli_exe_section](#) * section
- uint32_t offset
- uint32_t ep
- uint16_t nsections
- uint32_t res_addr
- uint32_t hdr_size

15.1.1 Detailed Description

Executable file information

[PE](#)

15.1.2 Field Documentation

15.1.2.1 uint32_t ep

Entrypoint of executable

15.1.2.2 uint32_t hdr_size

Address size - PE ONLY

15.1.2.3 uint16_t nsections

Number of sections

15.1.2.4 uint32_t offset

Offset where this executable start in file (nonzero if embedded)

15.1.2.5 uint32_t res_addr

Resources RVA - PE ONLY

15.1.2.6 struct cli_exe_section* section

Information about all the sections of this file. This array has `nsection` elements

15.2 cli_exe_section Struct Reference

Data Fields

- uint32_t [rva](#)
- uint32_t [vsz](#)
- uint32_t [raw](#)
- uint32_t [rsz](#)
- uint32_t [chr](#)
- uint32_t [urva](#)
- uint32_t [uvsz](#)
- uint32_t [uraw](#)
- uint32_t [ursz](#)

15.2.1 Detailed Description

Section of executable file.

PE

15.2.2 Field Documentation

15.2.2.1 uint32_t chr

Section characteristics

15.2.2.2 uint32_t raw

Raw offset (in file)

15.2.2.3 uint32_t rsz

Raw size (in file)

15.2.2.4 uint32_t rva

Relative VirtualAddress

15.2.2.5 uint32_t uraw

PE - unaligned PointerToRawData

15.2.2.6 uint32_t ursz

PE - unaligned SizeOfRawData

15.2.2.7 uint32_t urva

PE - unaligned VirtualAddress

15.2.2.8 uint32_t uvsz

PE - unaligned VirtualSize

15.2.2.9 uint32_t vsz

VirtualSize

15.3 cli_pe_hook_data Struct Reference

Data Fields

- uint32_t [ep](#)
- uint16_t [nsections](#)
- struct [pe_image_file_hdr](#) [file_hdr](#)
- struct [pe_image_optional_hdr32](#) [opt32](#)
- struct [pe_image_optional_hdr64](#) [opt64](#)
- struct [pe_image_data_dir](#) [dirs](#) [16]
- uint32_t [e_lfanew](#)
- uint32_t [overlays](#)
- int32_t [overlays_sz](#)
- uint32_t [hdr_size](#)

15.3.1 Detailed Description

Data for the bytecode PE hook

PE

15.3.2 Field Documentation

15.3.2.1 struct `pe_image_data_dir` `dirs`[16]

PE data directory header

15.3.2.2 `uint32_t` `e_lfanew`

address of new exe header

15.3.2.3 `uint32_t` `ep`

EntryPoint as file offset

15.3.2.4 struct `pe_image_file_hdr` `file_hdr`

Header for this PE file

15.3.2.5 `uint32_t` `hdr_size`

internally needed by `rawaddr`

15.3.2.6 `uint16_t` `nsections`

Number of sections

15.3.2.7 struct `pe_image_optional_hdr32` `opt32`

32-bit PE optional header

15.3.2.8 struct `pe_image_optional_hdr64` `opt64`

64-bit PE optional header

15.3.2.9 `uint32_t` `overlays`

number of overlays

15.3.2.10 `int32_t` `overlays_sz`

size of overlays

15.4 DIS_arg Struct Reference

Data Fields

- enum [DIS_ACCESS](#) `access_type`
- enum [DIS_SIZE](#) `access_size`
- struct [DIS_mem_arg](#) `mem`
- enum [X86REGS](#) `reg`
- `uint64_t` `other`

15.4.1 Detailed Description

disassembled operand

[Disassemble](#)

15.4.2 Field Documentation

15.4.2.1 enum DIS_SIZE access_size

size of access

15.4.2.2 enum DIS_ACCESS access_type

type of access

15.4.2.3 struct DIS_mem_arg mem

memory operand

15.4.2.4 uint64_t other

other operand

15.4.2.5 enum X86REGS reg

register operand

15.5 DIS_fixed Struct Reference

Data Fields

- enum [X86OPS x86_opcode](#)
- enum [DIS_SIZE operation_size](#)
- enum [DIS_SIZE address_size](#)
- uint8_t [segment](#)

15.5.1 Detailed Description

disassembled instruction.

[Disassemble](#)

15.5.2 Field Documentation

15.5.2.1 enum DIS_SIZE address_size

size of address

15.5.2.2 enum DIS_SIZE operation_size

size of operation

15.5.2.3 uint8_t segment

segment

15.5.2.4 enum X86OPS x86_opcode

opcode of X86 instruction

15.6 DIS_mem_arg Struct Reference

Data Fields

- enum [DIS_SIZE access_size](#)
- enum [X86REGS scale_reg](#)
- enum [X86REGS add_reg](#)
- uint8_t [scale](#)
- int32_t [displacement](#)

15.6.1 Detailed Description

disassembled memory operand: $\text{scale_reg} * \text{scale} + \text{add_reg} + \text{displacement}$

Disassemble

15.6.2 Field Documentation

15.6.2.1 enum DIS_SIZE access_size

size of access

15.6.2.2 enum X86REGS add_reg

register used as displacement

15.6.2.3 int32_t displacement

displacement as immediate number

15.6.2.4 uint8_t scale

scale as immediate number

15.6.2.5 enum X86REGS scale_reg

register used as scale

15.7 DISASM_RESULT Struct Reference

15.7.1 Detailed Description

disassembly result, 64-byte, matched by type-8 signatures

15.8 pe_image_data_dir Struct Reference

15.8.1 Detailed Description

PE data directory header

PE

15.9 pe_image_file_hdr Struct Reference

Data Fields

- uint32_t [Magic](#)
- uint16_t [Machine](#)
- uint16_t [NumberOfSections](#)
- uint32_t [TimeDateStamp](#)
- uint32_t [PointerToSymbolTable](#)
- uint32_t [NumberOfSymbols](#)
- uint16_t [SizeOfOptionalHeader](#)

15.9.1 Detailed Description

Header for this PE file

PE

15.9.2 Field Documentation

15.9.2.1 uint16_t Machine

CPU this executable runs on, see libclamav/pe.c for possible values

15.9.2.2 uint32_t Magic

PE magic header: PE\0\0

15.9.2.3 uint16_t NumberOfSections

Number of sections in this executable

15.9.2.4 uint32_t NumberOfSymbols

debug

15.9.2.5 uint32_t PointerToSymbolTable

debug

15.9.2.6 uint16_t SizeOfOptionalHeader

== 224

15.9.2.7 uint32_t TimeDateStamp

Unreliable

15.10 pe_image_optional_hdr32 Struct Reference

Data Fields

- uint8_t [MajorLinkerVersion](#)
- uint8_t [MinorLinkerVersion](#)
- uint32_t [SizeOfCode](#)
- uint32_t [SizeOfInitializedData](#)
- uint32_t [SizeOfUninitializedData](#)
- uint32_t [ImageBase](#)
- uint32_t [SectionAlignment](#)
- uint32_t [FileAlignment](#)
- uint16_t [MajorOperatingSystemVersion](#)
- uint16_t [MinorOperatingSystemVersion](#)
- uint16_t [MajorImageVersion](#)
- uint16_t [MinorImageVersion](#)
- uint32_t [Checksum](#)
- uint32_t [NumberOfRvaAndSizes](#)

15.10.1 Detailed Description

32-bit PE optional header

PE

15.10.2 Field Documentation

15.10.2.1 uint32_t CheckSum

NT drivers only

15.10.2.2 uint32_t FileAlignment

usually 32 or 512

15.10.2.3 uint32_t ImageBase

multiple of 64 KB

15.10.2.4 uint16_t MajorImageVersion

unreliable

15.10.2.5 uint8_t MajorLinkerVersion

unreliable

15.10.2.6 uint16_t MajorOperatingSystemVersion

not used

15.10.2.7 uint16_t MinorImageVersion

unreliable

15.10.2.8 uint8_t MinorLinkerVersion

unreliable

15.10.2.9 uint16_t MinorOperatingSystemVersion

not used

15.10.2.10 uint32_t NumberOfRvaAndSizes

unreliable

15.10.2.11 uint32_t SectionAlignment

usually 32 or 4096

15.10.2.12 uint32_t SizeOfCode

unreliable

15.10.2.13 uint32_t SizeOfInitializedData

unreliable

15.10.2.14 uint32_t SizeOfUninitializedData

unreliable

15.11 pe_image_optional_hdr64 Struct Reference

Data Fields

- uint8_t [MajorLinkerVersion](#)
- uint8_t [MinorLinkerVersion](#)
- uint32_t [SizeOfCode](#)
- uint32_t [SizeOfInitializedData](#)
- uint32_t [SizeOfUninitializedData](#)
- uint64_t [ImageBase](#)
- uint32_t [SectionAlignment](#)
- uint32_t [FileAlignment](#)
- uint16_t [MajorOperatingSystemVersion](#)
- uint16_t [MinorOperatingSystemVersion](#)
- uint16_t [MajorImageVersion](#)
- uint16_t [MinorImageVersion](#)
- uint32_t [Checksum](#)
- uint32_t [NumberOfRvaAndSizes](#)

15.11.1 Detailed Description

PE 64-bit optional header

PE

15.11.2 Field Documentation

15.11.2.1 uint32_t CheckSum

NT drivers only

15.11.2.2 uint32_t FileAlignment

usually 32 or 512

15.11.2.3 uint64_t ImageBase

multiple of 64 KB

15.11.2.4 uint16_t MajorImageVersion

unreliable

15.11.2.5 uint8_t MajorLinkerVersion

unreliable

15.11.2.6 uint16_t MajorOperatingSystemVersion

not used

15.11.2.7 uint16_t MinorImageVersion

unreliable

15.11.2.8 uint8_t MinorLinkerVersion

unreliable

15.11.2.9 uint16_t MinorOperatingSystemVersion

not used

15.11.2.10 uint32_t NumberOfRvaAndSizes

unreliable

15.11.2.11 uint32_t SectionAlignment

usually 32 or 4096

15.11.2.12 uint32_t SizeOfCode

unreliable

15.11.2.13 uint32_t SizeOfInitializedData

unreliable

15.11.2.14 uint32_t SizeOfUninitializedData

unreliable

15.12 pe_image_section_hdr Struct Reference

Data Fields

- uint8_t [Name](#) [8]
- uint32_t [SizeOfRawData](#)
- uint32_t [PointerToRawData](#)
- uint32_t [PointerToRelocations](#)
- uint32_t [PointerToLinenumbers](#)
- uint16_t [NumberOfRelocations](#)
- uint16_t [NumberOfLinenumbers](#)

15.12.1 Detailed Description

PE section header

PE

15.12.2 Field Documentation

15.12.2.1 uint8_t Name[8]

may not end with NULL

15.12.2.2 uint16_t NumberOfLinenumbers

object files only

15.12.2.3 uint16_t NumberOfRelocations

object files only

15.12.2.4 uint32_t PointerToLinenumbers

object files only

15.12.2.5 uint32_t PointerToRawData

offset to the section's data

15.12.2.6 uint32_t PointerToRelocations

object files only

15.12.2.7 uint32_t SizeOfRawData

multiple of FileAlignment

16 File Documentation

16.1 bytecode_api.h File Reference

Enumerations

- enum [BytecodeKind](#) { [BC_GENERIC](#) = 0 , [BC_LOGICAL](#) = 256, [BC_PE_UNPACKER](#) }
- enum { [PE_INVALID_RVA](#) = 0xFFFFFFFF }
- enum [FunctionalityLevels](#)
- enum [pdf_phase](#)
- enum [pdf_flag](#)
- enum [pdf_objflags](#)
- enum { [SEEK_SET](#) = 0, [SEEK_CUR](#), [SEEK_END](#) }

Functions

- uint32_t [test1](#) (uint32_t a, uint32_t b)
- int32_t [read](#) (uint8_t *data, int32_t size)
Reads specified amount of bytes from the current file into a buffer. Also moves current position in the file.
- int32_t [write](#) (uint8_t *data, int32_t size)
Writes the specified amount of bytes from a buffer to the current temporary file.
- int32_t [seek](#) (int32_t pos, uint32_t whence)
Changes the current file position to the specified one.
- uint32_t [setvirusname](#) (const uint8_t *name, uint32_t len)
- uint32_t [debug_print_str](#) (const uint8_t *str, uint32_t len)
- uint32_t [debug_print_uint](#) (uint32_t a)
- uint32_t [disasm_x86](#) (struct [DISASM_RESULT](#) *result, uint32_t len)
- uint32_t [pe_rawaddr](#) (uint32_t rva)
- int32_t [file_find](#) (const uint8_t *data, uint32_t len)
- int32_t [file_byteat](#) (uint32_t offset)
- void * [malloc](#) (uint32_t size)
- uint32_t [test2](#) (uint32_t a)
- int32_t [get_pe_section](#) (struct [cli_exe_section](#) *section, uint32_t num)
- int32_t [fill_buffer](#) (uint8_t *buffer, uint32_t len, uint32_t filled, uint32_t cursor, uint32_t fill)
- int32_t [extract_new](#) (int32_t id)
- int32_t [read_number](#) (uint32_t radix)
- int32_t [hashset_new](#) (void)
- int32_t [hashset_add](#) (int32_t hs, uint32_t key)
- int32_t [hashset_remove](#) (int32_t hs, uint32_t key)
- int32_t [hashset_contains](#) (int32_t hs, uint32_t key)
- int32_t [hashset_done](#) (int32_t id)
- int32_t [hashset_empty](#) (int32_t id)
- int32_t [buffer_pipe_new](#) (uint32_t size)

- `int32_t buffer_pipe_new_fromfile` (`uint32_t pos`)
- `uint32_t buffer_pipe_read_avail` (`int32_t id`)
- `uint8_t * buffer_pipe_read_get` (`int32_t id`, `uint32_t amount`)
- `int32_t buffer_pipe_read_stopped` (`int32_t id`, `uint32_t amount`)
- `uint32_t buffer_pipe_write_avail` (`int32_t id`)
- `uint8_t * buffer_pipe_write_get` (`int32_t id`, `uint32_t size`)
- `int32_t buffer_pipe_write_stopped` (`int32_t id`, `uint32_t amount`)
- `int32_t buffer_pipe_done` (`int32_t id`)
- `int32_t inflate_init` (`int32_t from_buffer`, `int32_t to_buffer`, `int32_t windowBits`)
- `int32_t inflate_process` (`int32_t id`)
- `int32_t inflate_done` (`int32_t id`)
- `int32_t bytecode_rt_error` (`int32_t locationid`)
- `int32_t jsnorm_init` (`int32_t from_buffer`)
- `int32_t jsnorm_process` (`int32_t id`)
- `int32_t jsnorm_done` (`int32_t id`)
- `int32_t ilog2` (`uint32_t a`, `uint32_t b`)
- `int32_t ipow` (`int32_t a`, `int32_t b`, `int32_t c`)
- `uint32_t iexp` (`int32_t a`, `int32_t b`, `int32_t c`)
- `int32_t isin` (`int32_t a`, `int32_t b`, `int32_t c`)
- `int32_t icos` (`int32_t a`, `int32_t b`, `int32_t c`)
- `int32_t memstr` (`const uint8_t *haystack`, `int32_t haysize`, `const uint8_t *needle`, `int32_t needlesize`)
- `int32_t hex2ui` (`uint32_t hex1`, `uint32_t hex2`)
- `int32_t atoi` (`const uint8_t *str`, `int32_t size`)
- `uint32_t debug_print_str_start` (`const uint8_t *str`, `uint32_t len`)
- `uint32_t debug_print_str_nonl` (`const uint8_t *str`, `uint32_t len`)
- `uint32_t entropy_buffer` (`uint8_t *buffer`, `int32_t size`)
- `int32_t map_new` (`int32_t keysize`, `int32_t valuesize`)
- `int32_t map_addkey` (`const uint8_t *key`, `int32_t ksize`, `int32_t id`)
- `int32_t map_setvalue` (`const uint8_t *value`, `int32_t vsize`, `int32_t id`)
- `int32_t map_remove` (`const uint8_t *key`, `int32_t ksize`, `int32_t id`)
- `int32_t map_find` (`const uint8_t *key`, `int32_t ksize`, `int32_t id`)
- `int32_t map_getvaluesize` (`int32_t id`)
- `uint8_t * map_getvalue` (`int32_t id`, `int32_t size`)
- `int32_t map_done` (`int32_t id`)
- `int32_t file_find_limit` (`const uint8_t *data`, `uint32_t len`, `int32_t maxpos`)
- `uint32_t engine_functionality_level` (`void`)
- `uint32_t engine_dconf_level` (`void`)
- `uint32_t engine_scan_options` (`void`)
- `uint32_t engine_db_options` (`void`)
- `int32_t extract_set_container` (`uint32_t container`)
- `int32_t input_switch` (`int32_t extracted_file`)
- `uint32_t get_environment` (`struct cli_environment *env`, `uint32_t len`)
- `uint32_t disable_bytecode_if` (`const int8_t *reason`, `uint32_t len`, `uint32_t cond`)
- `uint32_t disable_jit_if` (`const int8_t *reason`, `uint32_t len`, `uint32_t cond`)
- `int32_t version_compare` (`const uint8_t *lhs`, `uint32_t lhs_len`, `const uint8_t *rhs`, `uint32_t rhs_len`)

- uint32_t [check_platform](#) (uint32_t a, uint32_t b, uint32_t c)
- int32_t [pdf_get_obj_num](#) (void)
- int32_t [pdf_get_flags](#) (void)
- int32_t [pdf_set_flags](#) (int32_t flags)
- int32_t [pdf_lookupobj](#) (uint32_t id)
- uint32_t [pdf_getobjsize](#) (int32_t objidx)
- uint8_t * [pdf_getobj](#) (int32_t objidx, uint32_t amount)
- int32_t [pdf_get_phase](#) (void)
- int32_t [pdf_get_dumpedobjid](#) (void)
- int32_t [matchicon](#) (const uint8_t *group1, int32_t group1_len, const uint8_t *group2, int32_t group2_len)

Variables

- const uint32_t [__clambc_match_counts](#) [64]
Logical signature match counts.
- const uint32_t [__clambc_match_offsets](#) [64]
Logical signature match offsets This is a low-level variable, use the Macros in [bytecode_local.h](#) instead to access it.
- struct [cli_pe_hook_data](#) [__clambc_pedata](#)
- const uint32_t [__clambc_filesize](#) [1]
- const uint16_t [__clambc_kind](#)

16.1.1 Detailed Description

16.1.2 Enumeration Type Documentation

16.1.2.1 anonymous enum

Enumerator:

PE_INVALID_RVA Invalid RVA specified

16.1.2.2 anonymous enum

Enumerator:

SEEK_SET set file position to specified absolute position

SEEK_CUR set file position relative to current position

SEEK_END set file position relative to file end

16.1.2.3 enum BytecodeKind

Bytecode trigger kind

Enumerator:

BC_GENERIC generic bytecode, not tied a specific hook

BC_LOGICAL triggered by a logical signature

BC_PE_UNPACKER a PE unpacker

16.1.2.4 enum **FunctionalityLevels**

LibClamAV functionality level constants

16.1.2.5 enum **pdf_flag**

PDF flags

16.1.2.6 enum **pdf_objflags**

PDF obj flags

16.1.2.7 enum **pdf_phase**

Phase of PDF parsing

16.1.3 Function Documentation

16.1.3.1 `int32_t atoi (const uint8_t * str, int32_t size)`

Converts string to positive number.

Parameters

<i>str</i>	buffer
<i>size</i>	size of <i>str</i>

Returns

>0 string converted to number if possible, -1 on error

String operation

16.1.3.2 `int32_t buffer_pipe_done (int32_t id)`

Deallocate memory used by buffer.

Data structure

After this all attempts to use this buffer will result in error. All `buffer_pipes` are automatically deallocated when bytecode finishes execution.

Parameters

<i>id</i>	ID of <code>buffer_pipe</code>
-----------	--------------------------------

Returns

0 on success

16.1.3.3 `int32_t buffer_pipe_new (uint32_t size)`

Creates a new pipe with the specified buffer size

Data structure**Parameters**

<i>size</i>	size of buffer
-------------	----------------

Returns

ID of newly created buffer_pipe

16.1.3.4 `int32_t buffer_pipe_new_fromfile (uint32_t pos)`

Same as buffer_pipe_new, except the pipe's input is tied

Data structure**File operation**

to the current file, at the specified position.

Parameters

<i>pos</i>	starting position of pipe input in current file
------------	---

Returns

ID of newly created buffer_pipe

16.1.3.5 `uint32_t buffer_pipe_read_avail (int32_t id)`

Returns the amount of bytes available to read.

Data structure**Parameters**

<i>id</i>	ID of buffer_pipe
-----------	-------------------

Returns

amount of bytes available to read

16.1.3.6 `uint8_t* buffer_pipe_read_get (int32_t id, uint32_t amount)`

Returns a pointer to the buffer for reading.

Data structure

The 'amount' parameter should be obtained by a call to [buffer_pipe_read_avail\(\)](#).

Parameters

<i>id</i>	ID of buffer_pipe
<i>amount</i>	to read

Returns

pointer to buffer, or NULL if buffer has less than specified amount

16.1.3.7 `int32_t buffer_pipe_read_stopped (int32_t id, uint32_t amount)`

Data structure

Updates read cursor in buffer_pipe.

Parameters

<i>id</i>	ID of buffer_pipe
<i>amount</i>	amount of bytes to move read cursor

Returns

0 on success

16.1.3.8 `uint32_t buffer_pipe_write_avail (int32_t id)`

Returns the amount of bytes available for writing.

Data structure**Parameters**

<i>id</i>	ID of buffer_pipe
-----------	-------------------

Returns

amount of bytes available for writing

16.1.3.9 `uint8_t* buffer_pipe_write_get (int32_t id, uint32_t size)`

Data structure

Returns pointer to writable buffer. The 'amount' parameter should be obtained by a call to [buffer_pipe_write_avail\(\)](#).

Parameters

<i>id</i>	ID of buffer_pipe
<i>size</i>	amount of bytes to write

Returns

pointer to write buffer, or NULL if requested amount is more than what is available in the buffer

16.1.3.10 `int32_t buffer_pipe_write_stopped (int32_t id, uint32_t amount)`

Updates the write cursor in buffer_pipe.

Data structure

Parameters

<i>id</i>	ID of buffer_pipe
<i>amount</i>	amount of bytes to move write cursor

Returns

0 on success

16.1.3.11 `int32_t bytecode_rt_error (int32_t locationid)`

Report a runtime error at the specified locationID.

Scan

Parameters

<i>locationid</i>	(line << 8) (column&0xff)
-------------------	-----------------------------

Returns

0

16.1.3.12 `uint32_t check_platform (uint32_t a, uint32_t b, uint32_t c)`

Disables the JIT if the platform id matches. 0xff can be used instead of a field to mark ANY.

Parameters

<i>a</i>	- os_category << 24 arch << 20 compiler << 16 flevel << 8 dconf
<i>b</i>	- big_endian << 28 sizeof_ptr << 24 cpp_version
<i>c</i>	- os_features << 24 c_version

Returns

0 - no match 1 - match

Environment

16.1.3.13 `uint32_t debug_print_str (const uint8_t * str, uint32_t len)`

Prints a debug message.

Parameters

<i>in</i>	<i>str</i>	Message to print
<i>in</i>	<i>len</i>	length of message to print

Returns

0

String operation

16.1.3.14 `uint32_t debug_print_str_nonl (const uint8_t * str, uint32_t len)`

Prints a debug message with a trailing newline, and not preceded by 'LibClamAV debug'.

Parameters

<i>str</i>	the string
<i>len</i>	length of <i>str</i>

Returns

0

String operation

16.1.3.15 `uint32_t debug_print_str_start (const uint8_t * str, uint32_t len)`

Prints a debug message with a trailing newline, but preceded by 'LibClamAV debug'.

Parameters

<i>str</i>	the string
<i>len</i>	length of <i>str</i>

Returns

0

String operation**16.1.3.16** `uint32_t debug_print_uint (uint32_t a)`

Prints a number as a debug message. This is like `debug_print_str_nonl!`

Parameters

<i>in</i>	<i>a</i>	number to print
-----------	----------	-----------------

Returns

0

String operation**16.1.3.17** `uint32_t disable_bytecode_if (const int8_t * reason, uint32_t len, uint32_t cond)`

Disables the bytecode completely if condition is true. Can only be called from the BC_STARTUP bytecode.

Parameters

<i>reason</i>	- why the bytecode had to be disabled
<i>len</i>	- length of reason
<i>cond</i>	- condition

Returns

0 - auto mode 1 - JIT disabled 2 - fully disabled

Environment**16.1.3.18** `uint32_t disable_jit_if (const int8_t * reason, uint32_t len, uint32_t cond)`

Disables the JIT completely if condition is true. Can only be called from the BC_STARTUP bytecode.

Parameters

<i>reason</i>	- why the JIT had to be disabled
<i>len</i>	- length of reason
<i>cond</i>	- condition

Returns

0 - auto mode 1 - JIT disabled 2 - fully disabled

Environment

16.1.3.19 `uint32_t disasm_x86 (struct DISASM_RESULT * result, uint32_t len)`

Disassembles starting from current file position, the specified amount of bytes.

Parameters

<i>out</i>	<i>result</i>	pointer to struct holding result
<i>in</i>	<i>len</i>	how many bytes to disassemble

Returns

0 for success

You can use `lseek` to disassemble starting from a different location. This is a low-level API, the result is in ClamAV type-8 signature format (64 bytes/instruction).

See also

[DisassembleAt](#)

Disassemble

16.1.3.20 `uint32_t engine_db_options (void)`

Returns the current engine's db options.

Returns

CL_DB_* flags

Engine query

16.1.3.21 `uint32_t engine_dconf_level (void)`

Returns the current engine (dconf) functionality level. Usually identical to [engine_functionality_level\(\)](#), unless distro backported patches. Compare with [FunctionalityLevels](#).

Returns

an integer representing the DCONF (security fixes) level.

Engine query**16.1.3.22 uint32_t engine_functionality_level (void)**

Returns the current engine (feature) functionality level. To map these to ClamAV releases, compare it with [FunctionalityLevels](#).

Returns

an integer representing current engine functionality level.

Engine query**16.1.3.23 uint32_t engine_scan_options (void)**

Returns the current engine's scan options.

Returns

CL_SCAN* flags

Engine query**16.1.3.24 uint32_t entropy_buffer (uint8_t * buffer, int32_t size)**

Returns an approximation for the entropy of *buffer*.

Parameters

<i>buffer</i>	input buffer
<i>size</i>	size of buffer

Returns

entropy estimation * 2²⁶

String operation

16.1.3.25 int32_t extract_new (int32_t *id*)

Prepares for extracting a new file, if we've already extracted one it scans it.

Scan

Parameters

<i>in</i>	<i>id</i>	an id for the new file (for example position in container)
-----------	-----------	--

Returns

1 if previous extracted file was infected

16.1.3.26 int32_t extract_set_container (uint32_t *container*)

Sets the container type for the currently extracted file.

Parameters

<i>container</i>	container type (CL_TYPE_*)
------------------	----------------------------

Returns

current setting for container (CL_TYPE_ANY default)

Scan

16.1.3.27 int32_t file_bytest (uint32_t *offset*)

Read a single byte from current file

File operation

Parameters

<i>offset</i>	file offset
---------------	-------------

Returns

byte at offset *offset* in the current file, or -1 if offset is invalid

16.1.3.28 int32_t file_find (const uint8_t * *data*, uint32_t *len*)

Looks for the specified sequence of bytes in the current file.

File operation

Parameters

in	<i>data</i>	the sequence of bytes to look for
	<i>len</i>	length of <i>data</i> , cannot be more than 1024

Returns

offset in the current file if match is found, -1 otherwise

16.1.3.29 `int32_t file_find_limit (const uint8_t * data, uint32_t len, int32_t maxpos)`

Looks for the specified sequence of bytes in the current file, up to the specified position.

Parameters

in	<i>data</i>	the sequence of bytes to look for
	<i>len</i>	length of <i>data</i> , cannot be more than 1024
	<i>maxpos</i>	maximum position to look for a match, note that this is 1 byte after the end of last possible match: <i>match_pos</i> + <i>len</i> < <i>maxpos</i>

Returns

offset in the current file if match is found, -1 otherwise

File operation

16.1.3.30 `int32_t fill_buffer (uint8_t * buffer, uint32_t len, uint32_t filled, uint32_t cursor, uint32_t fill)`

Fills the specified buffer with at least *fill* bytes.

File operation

Parameters

out	<i>buffer</i>	the buffer to fill
in	<i>len</i>	length of buffer
in	<i>filled</i>	how much of the buffer is currently filled
in	<i>cursor</i>	position of cursor in buffer
in	<i>fill</i>	amount of bytes to fill in (0 is valid)

Returns

<0 on error, 0 on EOF, number bytes available in buffer (starting from 0) The character at the cursor will be at position 0 after this call.

16.1.3.31 uint32_t get_environment (struct cli_environment * *env*, uint32_t *len*)

Queries the environment this bytecode runs in. Used by BC_STARTUP to disable bytecode when bugs are known for the current platform.

Parameters

out	<i>env</i>	- the full environment
	<i>len</i>	- size of <i>env</i>

Returns

0

Environment

16.1.3.32 int32_t get_pe_section (struct cli_exe_section * *section*, uint32_t *num*)

Gets information about the specified PE section.

PE

Parameters

out	<i>section</i>	PE section information will be stored here
in	<i>num</i>	PE section number

Returns

0 - success -1 - failure

16.1.3.33 int32_t hashset_add (int32_t *hs*, uint32_t *key*)

Add a new 32-bit key to the hashset.

Data structure

Parameters

	<i>hs</i>	ID of hashset (from hashset_new)
	<i>key</i>	the key to add

Returns

0 on success

16.1.3.34 int32_t hashset_contains (int32_t *hs*, uint32_t *key*)

Returns whether the hashset contains the specified key.

Data structure

Parameters

<i>hs</i>	ID of hashset (from hashset_new)
<i>key</i>	the key to lookup

Returns

1 if found, 0 if not found, <0 on invalid hashset ID

16.1.3.35 int32_t hashset_done (int32_t *id*)

Deallocates the memory used by the specified hashset.

Data structure

Trying to use the hashset after this will result in an error. The hashset may not be used after this. All hashsets are automatically deallocated when bytecode finishes execution.

Parameters

<i>id</i>	ID of hashset (from hashset_new)
-----------	----------------------------------

Returns

0 on success

16.1.3.36 int32_t hashset_empty (int32_t *id*)

Returns whether the hashset is empty.

Data structure

Parameters

<i>id</i>	of hashset (from hashset_new)
-----------	-------------------------------

Returns

0 on success

16.1.3.37 int32_t hashset_new (void)

Creates a new hashset and returns its id.

Data structure

Returns

ID for new hashset

16.1.3.38 `int32_t hashset_remove (int32_t hs, uint32_t key)`

Remove a 32-bit key from the hashset.

Data structure

Parameters

<i>hs</i>	ID of hashset (from <code>hashset_new</code>)
<i>key</i>	the key to add

Returns

0 on success

16.1.3.39 `int32_t hex2ui (uint32_t hex1, uint32_t hex2)`

Returns hexadecimal characters `hex1` and `hex2` converted to 8-bit number.

Parameters

<i>hex1</i>	hexadecimal character
<i>hex2</i>	hexadecimal character

Returns

`hex1 hex2` converted to 8-bit integer, -1 on error

String operation

16.1.3.40 `int32_t icos (int32_t a, int32_t b, int32_t c)`

Returns $c \cdot \cos(a/b)$.

Parameters

<i>a</i>	integer
<i>b</i>	integer
<i>c</i>	integer

Returns $c * \sin(a/b)$ **Math function****16.1.3.41** uint32_t iexp (int32_t *a*, int32_t *b*, int32_t *c*)Returns $\exp(a/b) * c$ **Parameters**

<i>a</i>	integer
<i>b</i>	integer
<i>c</i>	integer

Returns $c * \exp(a/b)$ **Math function****16.1.3.42** int32_t ilog2 (uint32_t *a*, uint32_t *b*)Returns $2^{26 * \log_2(a/b)}$ **Parameters**

<i>a</i>	input
<i>b</i>	input

Returns $2^{26 * \log_2(a/b)}$ **Math function****16.1.3.43** int32_t inflate_done (int32_t *id*)

Deallocates inflate data structure. Using the inflate data structure after this will result in an error. All inflate data structures are automatically deallocated when bytecode finishes execution.

Data structure

Parameters

<i>id</i>	ID of inflate data structure
-----------	------------------------------

Returns

0 on success.

16.1.3.44 `int32_t inflate_init (int32_t from_buffer, int32_t to_buffer, int32_t windowBits)`

Initializes inflate data structures for decompressing data

Data structure

'from_buffer' and writing uncompressed data 'to_buffer'.

Parameters

<i>from_buffer</i>	ID of buffer_pipe to read compressed data from
<i>to_buffer</i>	ID of buffer_pipe to write decompressed data to
<i>windowBits</i>	(see zlib documentation)

Returns

ID of newly created inflate data structure, <0 on failure

16.1.3.45 `int32_t inflate_process (int32_t id)`

Inflate all available data in the input buffer, and write to output buffer. Stops when the input buffer becomes empty, or write buffer becomes full. Also attempts to recover from corrupted inflate stream (via `inflateSync`). This function can be called repeatedly on success after filling the input buffer, and flushing the output buffer. The inflate stream is done processing when 0 bytes are available from output buffer, and input buffer is not empty.

Data structure**Parameters**

<i>id</i>	ID of inflate data structure
-----------	------------------------------

Returns

0 on success, zlib error code otherwise

16.1.3.46 `int32_t input_switch (int32_t extracted_file)`

Toggles the read/seek API to read from the currently extracted file, and back. You must call `seek` after switching inputs to position the cursor to a valid position.

Parameters

<i>extracted_</i> <i>file</i>	1 - switch to reading from extracted file, 0 - switch back to original input
----------------------------------	--

Returns

-1 on error (if no extracted file exists) 0 on success

Scan

16.1.3.47 `int32_t ipow (int32_t a, int32_t b, int32_t c)`

Returns $c*a^b$.

Parameters

<i>a</i>	integer
<i>b</i>	integer
<i>c</i>	integer

Returns

$c*\text{pow}(a,b)$

Math function

16.1.3.48 `int32_t isin (int32_t a, int32_t b, int32_t c)`

Returns $c*\sin(a/b)$.

Parameters

<i>a</i>	integer
<i>b</i>	integer
<i>c</i>	integer

Returns

$c*\sin(a/b)$

Math function

16.1.3.49 `int32_t jsnorm_done (int32_t id)`

Flushes JS normalizer.

JavaScript

Parameters

<i>id</i> ID of js normalizer to flush
--

Returns

0 - success -1 - failure

16.1.3.50 `int32_t jsnorm_init (int32_t from_buffer)`

Initializes JS normalizer for reading 'from_buffer'. Normalized JS will be written to a single tempfile, one normalized JS per line, and automatically scanned when the bytecode finishes execution.

JavaScript

Parameters

<i>from_buffer</i> ID of buffer_pipe to read javascript from
--

Returns

ID of JS normalizer, <0 on failure

16.1.3.51 `int32_t jsnorm_process (int32_t id)`

Normalize all javascript from the input buffer, and write to tempfile. You can call this function repeatedly on success, if you (re)fill the input buffer.

JavaScript

Parameters

<i>id</i> ID of JS normalizer

Returns

0 on success, <0 on failure

16.1.3.52 `void* malloc (uint32_t size)`

Allocates memory. Currently this memory is freed automatically on exit from the bytecode, and there is no way to free it sooner.

Data structure

Parameters

<i>size</i>	amount of memory to allocate in bytes
-------------	---------------------------------------

Returns

pointer to allocated memory

16.1.3.53 `int32_t map_addkey (const uint8_t * key, int32_t ksize, int32_t id)`

Inserts the specified key/value pair into the map.

Parameters

<i>id</i>	id of table
<i>key</i>	key
<i>ksize</i>	size of key

Returns

0 - if key existed before 1 - if key didn't exist before <0 - if ksize doesn't match
keysize specified at table creation

Data structure

16.1.3.54 `int32_t map_done (int32_t id)`

Deallocates the memory used by the specified map. Trying to use the map after this will result in an error. All maps are automatically deallocated when the bytecode finishes execution.

Parameters

<i>id</i>	id of map
-----------	-----------

Returns

0 - success -1 - invalid map

Data structure

16.1.3.55 `int32_t map_find (const uint8_t * key, int32_t ksize, int32_t id)`

Looks up key in map. The map remember the last looked up key (so you can retrieve the value).

Parameters

<i>id</i>	id of map
<i>key</i>	key
<i>ksize</i>	size of key

Returns

0 - if not found 1 - if found <0 - if ksize doesn't match the size specified at table creation

Data structure

16.1.3.56 `uint8_t* map_getvalue (int32_t id, int32_t size)`

Returns the value obtained during last `map_find`.

Parameters

<i>id</i>	id of map.
<i>size</i>	size of value (obtained from <code>map_getvaluesize</code>)

Returns

value

Data structure

16.1.3.57 `int32_t map_getvaluesize (int32_t id)`

Returns the size of value obtained during last `map_find`.

Parameters

<i>id</i>	id of map.
-----------	------------

Returns

size of value

Data structure

16.1.3.58 `int32_t map_new (int32_t keysize, int32_t valuesize)`

Creates a new map and returns its id.

Parameters

<i>keysize</i>	size of key
<i>valuesize</i>	size of value, if 0 then value is allocated separately

Returns

ID of new map

Data structure

16.1.3.59 `int32_t map_remove (const uint8_t * key, int32_t ksize, int32_t id)`

Remove an element from the map.

Parameters

<i>id</i>	id of map
<i>key</i>	key
<i>ksize</i>	size of key

Returns

0 on success, key was present 1 if key was not present <0 if ksize doesn't match
keysize specified at table creation

Data structure

16.1.3.60 `int32_t map_setvalue (const uint8_t * value, int32_t vsize, int32_t id)`

Sets the value for the last inserted key with `map_addkey`.

Parameters

<i>id</i>	id of table
<i>value</i>	value
<i>vsize</i>	size of value

Returns

0 - if update was successful <0 - if there is no last key

Data structure

16.1.3.61 `int32_t matchicon (const uint8_t * group1, int32_t group1_len, const uint8_t * group2,
int32_t group2_len)`

Attempts to match current executable's icon against the specified icon groups.

Icon

Parameters

<i>in</i>	<i>group1</i>	- same as GROUP1 in LDB signatures
-----------	---------------	------------------------------------

	<i>group1_len</i>	- length of <i>group1</i>
<i>in</i>	<i>group2</i>	- same as GROUP2 in LDB signatures
	<i>group2_len</i>	- length of <i>group2</i>

Returns

-1 - invalid call, or sizes (only valid for PE hooks) 0 - not a match 1 - match

16.1.3.62 `int32_t memstr (const uint8_t * haystack, int32_t haysize, const uint8_t * needle, int32_t needlesize)`

Return position of match, -1 otherwise.

Parameters

<i>haystack</i>	buffer to search
<i>haysize</i>	size of <i>haystack</i>
<i>needle</i>	substring to search
<i>needlesize</i>	size of <i>needle</i>

Returns

location of match, -1 otherwise

String operation

16.1.3.63 `int32_t pdf_get_dumpedobjid (void)`

Return the currently dumped obj index.

PDF

Valid only in PDF_PHASE_POSTDUMP.

Returns

>=0 - object index -1 - invalid phase

16.1.3.64 `int32_t pdf_get_flags (void)`

Return the flags for the entire PDF (as set so far).

Returns

-1 - if not called from PDF hook >=0 - pdf flags

PDF

16.1.3.65 `int32_t pdf_get_obj_num (void)`

Return number of pdf objects

Returns

-1 - if not called from PDF hook ≥ 0 - number of PDF objects

PDF

16.1.3.66 `int32_t pdf_get_phase (void)`

Return an 'enum pdf_phase'.

PDF

Identifies at which phase this bytecode was called.

Returns

the current `pdf_phase`

16.1.3.67 `uint8_t* pdf_getobj (int32_t objidx, uint32_t amount)`

Return the undecoded object.

PDF

Meant only for reading, write modifies the fmap buffer, so avoid!

Parameters

<i>objidx</i> - object index (from 0), not object id!
<i>amount</i> - size returned by <code>pdf_getobjsize</code> (or smaller)

Returns

NULL - invalid `objidx/amount` pointer - pointer to original object

16.1.3.68 `uint32_t pdf_getobjsize (int32_t objidx)`

Return the size of the specified PDF obj.

PDF

Parameters

<i>objidx</i> - object index (from 0), not object id!

Returns

0 - if not called from PDF hook, or invalid objnum ≥ 0 - size of object

16.1.3.69 `int32_t pdf_lookupobj (uint32_t id)`

Lookup pdf object with specified id.

PDF

Parameters

<i>id</i> - pdf id (objnumber \ll 8 generationid)

Returns

-1 - if object id doesn't exist ≥ 0 - object index

16.1.3.70 `int32_t pdf_set_flags (int32_t flags)`

Sets the flags for the entire PDF. It is recommended that you retrieve old flags, and just add new ones.

PDF

Parameters

<i>flags</i> - flags to set.

Returns

0 - success -1 - invalid phase

16.1.3.71 `uint32_t pe_rawaddr (uint32_t rva)`

Converts a RVA (Relative Virtual Address) to an absolute PE file offset.

Parameters

<i>rva</i> - a rva address from the PE file

Returns

absolute file offset mapped to the *rva*, or PE_INVALID_RVA if the *rva* is invalid.

PE

16.1.3.72 int32_t read (uint8_t * data, int32_t size)

Reads specified amount of bytes from the current file into a buffer. Also moves current position in the file.

Parameters

in	size	amount of bytes to read
out	data	pointer to buffer where data is read into

Returns

amount read.

File operation

16.1.3.73 int32_t read_number (uint32_t radix)

Reads a number in the specified radix starting from the current position.

File operation

Non-numeric characters are ignored.

Parameters

in	radix	10 or 16
----	-------	----------

Returns

the number read

16.1.3.74 int32_t seek (int32_t pos, uint32_t whence)

Changes the current file position to the specified one.

See also

[SEEK_SET](#), [SEEK_CUR](#), [SEEK_END](#)

Parameters

in	pos	offset (absolute or relative depending on whence param)
in	whence	one of SEEK_SET , SEEK_CUR , SEEK_END

Returns

absolute position in file

File operation

16.1.3.75 `uint32_t setvirusname (const uint8_t * name, uint32_t len)`

Sets the name of the virus found.

Parameters

in	<i>name</i>	the name of the virus
in	<i>len</i>	length of the virusname

Returns

0

Scan

16.1.3.76 `uint32_t test1 (uint32_t a, uint32_t b)`

Test api.

Parameters

	<i>a</i>	0xf00dbeef
	<i>b</i>	0xbeeff00d

Returns

0x12345678 if parameters match, 0x55 otherwise

16.1.3.77 `uint32_t test2 (uint32_t a)`

Test api2.

Parameters

	<i>a</i>	0xf00d
--	----------	--------

Returns

0xd00f if parameter matches, 0x5555 otherwise

16.1.3.78 `int32_t version_compare (const uint8_t * lhs, uint32_t lhs_len, const uint8_t * rhs, uint32_t rhs_len)`

Compares two version numbers.

Parameters

in	<i>lhs</i>	- left hand side of comparison
	<i>lhs_len</i>	- length of lhs
in	<i>rhs</i>	- right hand side of comparison
	<i>rhs_len</i>	- length of rhs

Returns

-1 - lhs < rhs 0 - lhs == rhs 1 - lhs > rhs

Environment

16.1.3.79 `int32_t write (uint8_t * data, int32_t size)`

Writes the specified amount of bytes from a buffer to the current temporary file.

Parameters

<i>in</i>	<i>data</i>	pointer to buffer of data to write
<i>in</i>	<i>size</i>	amount of bytes to write <i>size</i> bytes to temporary file, from the buffer pointed to byte

Returns

amount of bytes successfully written

File operation

16.1.4 Variable Documentation

16.1.4.1 `const uint32_t __clambc_filesize[1]`

File size (max 4G).

Global variable

16.1.4.2 `const uint16_t __clambc_kind`

Kind of the bytecode

Global variable

16.1.4.3 `const uint32_t __clambc_match_counts[64]`

Logical signature match counts.

This is a low-level variable, use the Macros in [bytecode_local.h](#) instead to access it.

Global variable

16.1.4.4 `const uint32_t __clambc_match_offsets[64]`

Logical signature match offsets This is a low-level variable, use the Macros in [bytecode_local.h](#) instead to access it.

Global variable16.1.4.5 `struct cli_pe_hook_data __clambc_pedata`

PE data, if this is a PE hook.

Global variable16.2 `bytecode_disasm.h` File Reference

Data Structures

- struct [DISASM_RESULT](#)

Enumerations

- enum [X86OPS](#) { ,
[OP_AAA](#), [OP_AAD](#), [OP_AAM](#), [OP_AAS](#),
[OP_ADD](#), [OP_ADC](#), [OP_AND](#), [OP_ARPL](#),
[OP_BOUND](#), [OP_BSF](#), [OP_BSR](#), [OP_BSWAP](#),
[OP_BT](#), [OP BTC](#), [OP_BTR](#), [OP BTS](#),
[OP_CALL](#), [OP_CDQ](#) , [OP_CWDE](#), [OP_CBW](#),
[OP_CLC](#), [OP_CLD](#), [OP_CLI](#), [OP_CLTS](#),
[OP_CMC](#), [OP_CMOVO](#), [OP_CMOVNO](#), [OP_CMOVC](#),
[OP_CMOVNC](#), [OP_CMOVZ](#), [OP_CMOVNZ](#), [OP_CMOVBE](#),
[OP_CMOVA](#), [OP_CMOVS](#), [OP_CMOVNS](#), [OP_CMOVP](#),
[OP_CMOVNP](#), [OP_CMOVL](#), [OP_CMOVGE](#), [OP_CMOVLE](#),
[OP_CMOVG](#), [OP_CMP](#), [OP_CMPSD](#), [OP_CMPSW](#),
[OP_CMPSB](#), [OP_CMPXCHG](#), [OP_CMPXCHG8B](#), [OP_CPUID](#),
[OP_DAA](#), [OP_DAS](#), [OP_DEC](#), [OP_DIV](#),
[OP_ENTER](#), [OP_FWAIT](#), [OP_HLT](#), [OP_IDIV](#),
[OP_IMUL](#), [OP_INC](#), [OP_IN](#), [OP_INSD](#),
[OP_INSW](#), [OP_INSB](#), [OP_INT](#) , [OP_INT3](#),
[OP_INT0](#), [OP_INVD](#) , [OP_INVLPG](#), [OP_IRET](#),

OP_JO, OP_JNO, OP_JC, OP_JNC,
OP_JZ, OP_JNZ, OP_JBE, OP_JA,
OP_JS, OP_JNS, OP_JP, OP_JNP,
OP_JL, OP_JGE, OP_JLE, OP_JG,
OP_JMP, OP_LAHF, OP_LAR, OP_LDS,
OP_LES, OP_LFS, OP_LGS, OP_LEA,
OP_LEAVE, OP_LGDT, OP_LIDT, OP_LLDT,
OP_PREFIX_LOCK, OP_LODSD, OP_LODSW, OP_LODSB,
OP_LOOP, OP_LOOPE, OP_LOOPNE, OP_JECXZ,
OP_LSL, OP_LSS, OP_LTR, OP_MOV,
OP_MOVSD, OP_MOVSW, OP_MOVSB, OP_MOVSX,
OP_MOVZX, OP_MUL, OP_NEG, OP_NOP,
OP_NOT, OP_OR, OP_OUT, OP_OUTSD,
OP_OUTSW, OP_OUTSB, OP_PUSH, OP_PUSHAD ,
OP_PUSHFD , OP_POP, OP_POPAD, OP_POPFD ,
OP_RCL, OP_RCR, OP_RDMSR, OP_RDPMC,
OP_RDTSC, OP_PREFIX_REPE, OP_PREFIX_REPN, OP_RETF,
OP_RETN, OP_ROL, OP_ROR, OP_RSM,
OP_SAHF, OP_SAR, OP_SBB, OP_SCASD,
OP_SCASW, OP_SCASB, OP_SETO, OP_SETNO,
OP_SETC, OP_SETNC, OP_SETZ, OP_SETNZ,
OP_SETBE, OP_SETA, OP_SETS, OP_SETNS,
OP_SETP, OP_SETNP, OP_SETL, OP_SETGE,
OP_SETLE, OP_SETG, OP_SGDT, OP_SIDT,
OP_SHL, OP_SHLD, OP_SHR, OP_SHRD,
OP_SLDT, OP_STOSD, OP_STOSW, OP_STOSB,
OP_STR, OP_STC, OP_STD, OP_STI,
OP_SUB, OP_SYSCALL, OP_SYSENTER, OP_SYSEXIT,
OP_SYSRET, OP_TEST, OP_UD2, OP_VERR,
OP_VERRW, OP_WBINVD, OP_WRMSR, OP_XADD,
OP_XCHG, OP_XLAT, OP_XOR , OP_FPU,
OP_F2XM1, OP_FABS, OP_FADD, OP_FADDP,
OP_FBLD, OP_FBSTP, OP_FCHS, OP_FCLEX,
OP_FCMOVB, OP_FCMOVBE, OP_FCMOVE, OP_FCMOVNB,
OP_FCMOVNBE, OP_FCMOVNE, OP_FCMOVNU, OP_FCMOVU,
OP_FCOM, OP_FCOMI, OP_FCOMIP, OP_FCOMP,
OP_FCOMPP, OP_FCOS, OP_FDECSTP, OP_FDIV,
OP_FDIVP, OP_FDIVR, OP_FDIVRP, OP_FFREE,

```

OP_FIADD, OP_FICOM, OP_FICOMP, OP_FIDIV,
OP_FIDIVR, OP_FILD, OP_FIMUL, OP_FINCSTP,
OP_FINIT, OP_FIST, OP_FISTP, OP_FISTTP,
OP_FISUB, OP_FISUBR, OP_FLD, OP_FLD1,
OP_FLDL2E, OP_FLDL2T,
OP_FLDLG2, OP_FLDLN2, OP_FLDPI, OP_FLDZ,
OP_FMUL, OP_FMULP, OP_FNOP, OP_FPATAN,
OP_FPREM, OP_FPREM1, OP_FPTAN, OP_FRNDINT,
OP_FRSTOR, OP_FSCALE, OP_FSINCOS, OP_FSQRT,
OP_FSAVE, OP_FST, OP_FSTCW, OP_FSTENV,
OP_FSTP, OP_FSTSW, OP_FSUB, OP_FSUBP,
OP_FSUBR, OP_FSUBRP, OP_FTST, OP_FUCOM,
OP_FUCOMI, OP_FUCOMIP, OP_FUCOMP, OP_FUCOMPP,
OP_FXAM, OP_FXCH, OP_FXTRACT, OP_FYL2X,
OP_FYL2XP1 }
• enum DIS_ACCESS {
    ACCESS_NOARG, ACCESS_IMM, ACCESS_REL, ACCESS_REG,
    ACCESS_MEM }
• enum DIS_SIZE {
    SIZEB, SIZEW, SIZED, SIZEF,
    SIZEQ, SIZET, SIZEPTR }
• enum X86REGS

```

16.2.1 Detailed Description

16.2.2 Enumeration Type Documentation

16.2.2.1 enum DIS_ACCESS

Access type

Enumerator:

```

ACCESS_NOARG  arg not present
ACCESS_IMM   immediate
ACCESS_REL   +/- immediate
ACCESS_REG   register
ACCESS_MEM   [memory]

```

16.2.2.2 enum DIS_SIZE

for mem access, immediate and relative

Enumerator:

- SIZEB** Byte size access
- SIZEW** Word size access
- SIZED** Doubleword size access
- SIZEF** 6-byte access (seg+reg pair)
- SIZEQ** Quadword access
- SIZET** 10-byte access
- SIZEPTR** ptr

16.2.2.3 enum X86OPS

X86 opcode

Enumerator:

- OP_AAA** Ascii Adjust after Addition
- OP_AAD** Ascii Adjust AX before Division
- OP_AAM** Ascii Adjust AX after Multiply
- OP_AAS** Ascii Adjust AL after Subtraction
- OP_ADD** Add
- OP_ADC** Add with Carry
- OP_AND** Logical And
- OP_ARPL** Adjust Requested Privilege Level
- OP_BOUND** Check Array Index Against Bounds
- OP_BSF** Bit Scan Forward
- OP_BSR** Bit Scan Reverse
- OP_BSWAP** Byte Swap
- OP_BT** Bit Test
- OPBTC** Bit Test and Complement
- OPBTR** Bit Test and Reset
- OPBTS** Bit Test and Set
- OP_CALL** Call
- OP_CDQ** Convert DoubleWord to QuadWord
- OP_CWDE** Convert Word to DoubleWord
- OP_CBW** Convert Byte to Word
- OP_CLC** Clear Carry Flag
- OP_CLD** Clear Direction Flag

OP_CLI Clear Interrupt Flag

OP_CLTS Clear Task-Switched Flag in CR0

OP_CMC Complement Carry Flag

OP_CMOVO Conditional Move if Overflow

OP_CMOVNO Conditional Move if Not Overflow

OP_CMOVNC Conditional Move if Carry

OP_CMOVNC Conditional Move if Not Carry

OP_CMOVZ Conditional Move if Zero

OP_CMOVNZ Conditional Move if Non-Zero

OP_CMOVBE Conditional Move if Below or Equal

OP_CMOVA Conditional Move if Above

OP_CMOVS Conditional Move if Sign

OP_CMOVNS Conditional Move if Not Sign

OP_CMOVP Conditional Move if Parity

OP_CMOVNP Conditional Move if Not Parity

OP_CMOVL Conditional Move if Less

OP_CMOVGE Conditional Move if Greater or Equal

OP_CMOVLE Conditional Move if Less than or Equal

OP_CMOVG Conditional Move if Greater

OP_CMP Compare

OP_CMPSD Compare String DoubleWord

OP_CMPSW Compare String Word

OP_CMPSB Compare String Byte

OP_CMPXCHG Compare and Exchange

OP_CMPXCHG8B Compare and Exchange Bytes

OP_CPUID CPU Identification

OP_DAA Decimal Adjust AL after Addition

OP_DAS Decimal Adjust AL after Subtraction

OP_DEC Decrement by 1

OP_DIV Unsigned Divide

OP_ENTER Make Stack Frame for Procedure Parameters

OP_FWAIT Wait

OP_HLT Halt

OP_IDIV Signed Divide

OP_IMUL Signed Multiply

OP_INC Increment by 1

OP_IN INput from port

OP_INSD INput from port to String Doubleword

OP_INSW INput from port to String Word
OP_INSB INput from port to String Byte
OP_INT INTerrupt
OP_INT3 INTerrupt 3 (breakpoint)
OP_INT0 INTerrupt 4 if Overflow
OP_INVD Invalidate Internal Caches
OP_INVLPG Invalidate TLB Entry
OP_IRET Interrupt Return
OP_JO Jump if Overflow
OP_JNO Jump if Not Overflow
OP_JC Jump if Carry
OP_JNC Jump if Not Carry
OP_JZ Jump if Zero
OP_JNZ Jump if Not Zero
OP_JBE Jump if Below or Equal
OP_JA Jump if Above
OP_JS Jump if Sign
OP_JNS Jump if Not Sign
OP_JP Jump if Parity
OP_JNP Jump if Not Parity
OP_JL Jump if Less
OP_JGE Jump if Greater or Equal
OP_JLE Jump if Less or Equal
OP_JG Jump if Greater
OP_JMP Jump (unconditional)
OP_LAHF Load Status Flags into AH Register
OP_LAR load Access Rights Byte
OP_LDS Load Far Pointer into DS
OP_LES Load Far Pointer into ES
OP_LFS Load Far Pointer into FS
OP_LGS Load Far Pointer into GS
OP_LEA Load Effective Address
OP_LEAVE High Level Procedure Exit
OP_LGDT Load Global Descript Table Register
OP_LIDT Load Interrupt Descriptor Table Register
OP_LLDT Load Local Descriptor Table Register
OP_PREFIX_LOCK Assert LOCK# Signal Prefix
OP_LODSD Load String Dword

OP_LODSW Load String Word
OP_LODSB Load String Byte
OP_LOOP Loop According to ECX Counter
OP_LOOPE Loop According to ECX Counter and ZF=1
OP_LOOPNE Loop According to ECX Counter and ZF=0
OP_JECXZ Jump if ECX is Zero
OP_LSL Load Segment Limit
OP_LSS Load Far Pointer into SS
OP_LTR Load Task Register
OP_MOV Move
OP_MOVSD Move Data from String to String Doubleword
OP_MOVSW Move Data from String to String Word
OP_MOVSB Move Data from String to String Byte
OP_MOVSX Move with Sign-Extension
OP_MOVZX Move with Zero-Extension
OP_MUL Unsigned Multiply
OP_NEG Two's Complement Negation
OP_NOP No Operation
OP_NOT One's Complement Negation
OP_OR Logical Inclusive OR
OP_OUT Output to Port
OP_OUTSD Output String to Port Doubleword
OP_OUTSW Output String to Port Word
OP_OUTSB Output String to Port Bytes
OP_PUSH Push Onto the Stack
OP_PUSHAD Push All Double General Purpose Registers
OP_PUSHFD Push EFLAGS Register onto the Stack
OP_POP Pop a Value from the Stack
OP_POPAD Pop All Double General Purpose Registers from the Stack
OP_POPFD Pop Stack into EFLAGS Register
OP_RCL Rotate Carry Left
OP_RCR Rotate Carry Right
OP_RDMSR Read from Model Specific Register
OP_RDPMSR Read Performance Monitoring Counters
OP_RDTSC Read Time-Stamp Counter
OP_PREFIX_REPE Repeat String Operation Prefix while Equal
OP_PREFIX_REPNB Repeat String Operation Prefix while Not Equal
OP_RET Return from Far Procedure

OP_RETN Return from Near Procedure
OP_ROL Rotate Left
OP_ROR Rotate Right
OP_RSM Resume from System Management Mode
OP_SAHF Store AH into Flags
OP_SAR Shift Arithmetic Right
OP_SBB Subtract with Borrow
OP_SCASD Scan String Doubleword
OP_SCASW Scan String Word
OP_SCASB Scan String Byte
OP_SETO Set Byte on Overflow
OP_SETNO Set Byte on Not Overflow
OP_SETC Set Byte on Carry
OP_SETNC Set Byte on Not Carry
OP_SETZ Set Byte on Zero
OP_SETNZ Set Byte on Not Zero
OP_SETBE Set Byte on Below or Equal
OP_SETA Set Byte on Above
OP_SETS Set Byte on Sign
OP_SETNS Set Byte on Not Sign
OP_SETP Set Byte on Parity
OP_SETNP Set Byte on Not Parity
OP_SETL Set Byte on Less
OP_SETGE Set Byte on Greater or Equal
OP_SETLE Set Byte on Less or Equal
OP_SETG Set Byte on Greater
OP_SGDT Store Global Descriptor Table Register
OP_SIDT Store Interrupt Descriptor Table Register
OP_SHL Shift Left
OP_SHLD Double Precision Shift Left
OP_SHR Shift Right
OP_SHRD Double Precision Shift Right
OP_SLDT Store Local Descriptor Table Register
OP_STOSD Store String Doubleword
OP_STOSW Store String Word
OP_STOSB Store String Byte
OP_STR Store Task Register
OP_STC Set Carry Flag

OP_STD Set Direction Flag
OP_STI Set Interrupt Flag
OP_SUB Subtract
OP_SYSCALL Fast System Call
OP_SYSENTER Fast System Call
OP_SYSEXIT Fast Return from Fast System Call
OP_SYSRET Return from Fast System Call
OP_TEST Logical Compare
OP_UD2 Undefined Instruction
OP_VERR Verify a Segment for Reading
OP_VERRW Verify a Segment for Writing
OP_WBINVD Write Back and Invalidate Cache
OP_WRMSR Write to Model Specific Register
OP_XADD Exchange and Add
OP_XCHG Exchange Register/Memory with Register
OP_XLAT Table Look-up Translation
OP_XOR Logical Exclusive OR
OP_FPU FPU operation
OP_F2XM1 Compute $2x^{-1}$
OP_FABS Absolute Value
OP_FADD Floating Point Add
OP_FADDP Floating Point Add, Pop
OP_FBLD Load Binary Coded Decimal
OP_FBSTP Store BCD Integer and Pop
OP_FCHS Change Sign
OP_FCLEX Clear Exceptions
OP_FCMOVB Floating Point Move on Below
OP_FCMOVBE Floating Point Move on Below or Equal
OP_FCMOVE Floating Point Move on Equal
OP_FCMOVNB Floating Point Move on Not Below
OP_FCMOVNBE Floating Point Move on Not Below or Equal
OP_FCMOVNE Floating Point Move on Not Equal
OP_FCMOVNU Floating Point Move on Not Unordered
OP_FCMOVU Floating Point Move on Unordered
OP_FCOM Compare Floating Pointer Values and Set FPU Flags
OP_FCOMI Compare Floating Pointer Values and Set EFLAGS
OP_FCOMIP Compare Floating Pointer Values and Set EFLAGS, Pop
OP_FCOMP Compare Floating Pointer Values and Set FPU Flags, Pop

OP_FCOMPP Compare Floating Pointer Values and Set FPU Flags, Pop Twice
OP_FCOS Cosine
OP_FDECSTP Decrement Stack Top Pointer
OP_FDIV Floating Point Divide
OP_FDIVP Floating Point Divide, Pop
OP_FDIVR Floating Point Reverse Divide
OP_FDIVRP Floating Point Reverse Divide, Pop
OP_FFREE Free Floating Point Register
OP_FIADD Floating Point Add
OP_FICOM Compare Integer
OP_FICOMP Compare Integer, Pop
OP_FIDIV Floating Point Divide by Integer
OP_FIDIVR Floating Point Reverse Divide by Integer
OP_FILD Load Integer
OP_FIMUL Floating Point Multiply with Integer
OP_FINCSTP Increment Stack-Top Pointer
OP_FINIT Initialize Floating-Point Unit
OP_FIST Store Integer
OP_FISTP Store Integer, Pop
OP_FISTTP Store Integer with Truncation
OP_FISUB Floating Point Integer Subtract
OP_FISUBR Floating Point Reverse Integer Subtract
OP_FLD Load Floating Point Value
OP_FLD1 Load Constant 1
OP_FLDCW Load x87 FPU Control Word
OP_FLDENV Load x87 FPU Environment
OP_FLDL2E Load Constant $\log_2(e)$
OP_FLDL2T Load Constant $\log_2(10)$
OP_FLDLG2 Load Constant $\log_{10}(2)$
OP_FLDLN2 Load Constant $\log_e(2)$
OP_FLDPI Load Constant PI
OP_FLDZ Load Constant Zero
OP_FMUL Floating Point Multiply
OP_FMULP Floating Point Multiply, Pop
OP_FNOP No Operation
OP_FPATAN Partial Arctangent
OP_FPREM Partial Remainder
OP_FPREM1 Partial Remainder

OP_FPTAN Partial Tangent
OP_FRNDINT Round to Integer
OP_FRSTOR Restore x86 FPU State
OP_FSCALE Scale
OP_FSINCOS Sine and Cosine
OP_FSQRT Square Root
OP_FSAVE Store x87 FPU State
OP_FST Store Floating Point Value
OP_FSTCW Store x87 FPU Control Word
OP_FSTENV Store x87 FPU Environment
OP_FSTP Store Floating Point Value, Pop
OP_FSTSW Store x87 FPU Status Word
OP_FSUB Floating Point Subtract
OP_FSUBP Floating Point Subtract, Pop
OP_FSUBR Floating Point Reverse Subtract
OP_FSUBRP Floating Point Reverse Subtract, Pop
OP_FTST Floating Point Test
OP_FUCOM Floating Point Unordered Compare
OP_FUCOMI Floating Point Unordered Compare with Integer
OP_FUCOMIP Floating Point Unorder Compare with Integer, Pop
OP_FUCOMP Floating Point Unorder Compare, Pop
OP_FUCOMPP Floating Point Unorder Compare, Pop Twice
OP_FXAM Examine ModR/M
OP_FXCH Exchange Register Contents
OP_FXTRACT Extract Exponent and Significand
OP_FYL2X Compute $y \cdot \log_2 x$
OP_FYL2XP1 Compute $y \cdot \log_2(x+1)$

16.2.2.4 `enum X86REGS`

X86 registers

16.3 `bytecode_execs.h` File Reference

Data Structures

- struct [cli_exe_section](#)
- struct [cli_exe_info](#)

16.3.1 Detailed Description

16.4 bytecode_local.h File Reference

Data Structures

- struct [DIS_mem_arg](#)
- struct [DIS_arg](#)
- struct [DIS_fixed](#)

Defines

- #define [VIRUSNAME_PREFIX](#)(name) const char __clambc_virusname_prefix[] = name;
- #define [VIRUSNAMES](#)(...) const char *const __clambc_virusnames[] = {__VA_ARGS__};
- #define [PE_UNPACKER_DECLARE](#) const uint16_t __clambc_kind = BC_PE_UNPACKER;
- #define [PDF_HOOK_DECLARE](#) const uint16_t __clambc_kind = BC_PDF;
- #define [BYTECODE_ABORT_HOOK](#) 0xcea5e
- #define [PE_HOOK_DECLARE](#) const uint16_t __clambc_kind = BC_PE_ALL;
- #define [SIGNATURES_DECL_BEGIN](#) struct __Signatures {
- #define [DECLARE_SIGNATURE](#)(name)
- #define [SIGNATURES_DECL_END](#) };
- #define [TARGET](#)(tgt) const unsigned short __Target = (tgt);
- #define [COPYRIGHT](#)(c) const char *const __Copyright = (c);
- #define [ICONGROUP1](#)(group) const char *const __IconGroup1 = (group);
- #define [ICONGROUP2](#)(group) const char *const __IconGroup2 = (group);
- #define [FUNCTIONALITY_LEVEL_MIN](#)(m) const unsigned short __FuncMin = (m);
- #define [FUNCTIONALITY_LEVEL_MAX](#)(m) const unsigned short __FuncMax = (m);
- #define [SIGNATURES_DEF_BEGIN](#)
- #define [DEFINE_SIGNATURE](#)(name, hex)
- #define [SIGNATURES_END](#) };\

Functions

- static force_inline void overloadable_func [debug](#) (const char *str)
- static force_inline void overloadable_func [debug](#) (const uint8_t *str)
- static force_inline void overloadable_func [debug](#) (uint32_t a)
- void [debug](#) (...) __attribute__((overloadable))
- static force_inline uint32_t [count_match](#) (__Signature sig)
- static force_inline uint32_t [matches](#) (__Signature sig)
- static force_inline uint32_t [match_location](#) (__Signature sig, uint32_t goback)

- static force_inline int32_t [match_location_check](#) (__Signature sig, uint32_t goback, const char *static_start, uint32_t static_len)
 - static force_inline overloadable_func void [foundVirus](#) (const char *virusname)
 - static force_inline void overloadable_func [foundVirus](#) (void)
 - static force_inline uint32_t [getFileSize](#) (void)
 - bool [__is_bigendian](#) (void) __attribute__((const)) __attribute__((nothrow))
 - static uint32_t force_inline [le32_to_host](#) (uint32_t v)
 - static uint64_t force_inline [le64_to_host](#) (uint64_t v)
 - static uint16_t force_inline [le16_to_host](#) (uint16_t v)
 - static uint32_t force_inline [cli_readint32](#) (const void *buff)
 - static uint16_t force_inline [cli_readint16](#) (const void *buff)
 - static void force_inline [cli_writeint32](#) (void *offset, uint32_t v)
 - static force_inline bool [hasExeInfo](#) (void)
 - static force_inline bool [hasPEInfo](#) (void)
 - static force_inline bool [isPE64](#) (void)
 - static static force_inline force_inline uint8_t [getPEMajorLinkerVersion](#) (void)
 - static force_inline uint8_t [getPEMinorLinkerVersion](#) (void)
 - static force_inline uint32_t [getPESizeOfCode](#) (void)
 - static force_inline uint32_t [getPESizeOfInitializedData](#) (void)
 - static force_inline uint32_t [getPESizeOfUninitializedData](#) (void)
 - static force_inline uint32_t [getPEBaseOfCode](#) (void)
 - static force_inline uint32_t [getPEBaseOfData](#) (void)
 - static force_inline uint64_t [getPEImageBase](#) (void)
 - static force_inline uint32_t [getPESectionAlignment](#) (void)
 - static force_inline uint32_t [getPEFileAlignment](#) (void)
 - static force_inline uint16_t [getPEMajorOperatingSystemVersion](#) (void)
 - static force_inline uint16_t [getPEMinorOperatingSystemVersion](#) (void)
 - static force_inline uint16_t [getPEMajorImageVersion](#) (void)
 - static force_inline uint16_t [getPEMinorImageVersion](#) (void)
 - static force_inline uint16_t [getPEMajorSubsystemVersion](#) (void)
 - static force_inline uint16_t [getPEMinorSubsystemVersion](#) (void)
 - static force_inline uint32_t [getPEWin32VersionValue](#) (void)
 - static force_inline uint32_t [getPESizeOfImage](#) (void)
 - static force_inline uint32_t [getPESizeOfHeaders](#) (void)
 - static force_inline uint32_t [getPEChecksum](#) (void)
 - static force_inline uint16_t [getPESubsystem](#) (void)
 - static force_inline uint16_t [getPEDllCharacteristics](#) (void)
- Return the PE DllCharacteristics.*
- static force_inline uint32_t [getPESizeOfStackReserve](#) (void)
 - static force_inline uint32_t [getPESizeOfStackCommit](#) (void)
 - static force_inline uint32_t [getPESizeOfHeapReserve](#) (void)
 - static force_inline uint32_t [getPESizeOfHeapCommit](#) (void)
 - static force_inline uint32_t [getPELoaderFlags](#) (void)
 - static force_inline uint16_t [getPEMachine](#) ()
 - static force_inline uint32_t [getPETimeDateStamp](#) ()
 - static force_inline uint32_t [getPEPointerToSymbolTable](#) ()

- static force_inline uint32_t [getPENumberOfSymbols](#) ()
- static force_inline uint16_t [getPESizeOfOptionalHeader](#) ()
- static force_inline uint16_t [getPECharacteristics](#) ()
- static force_inline bool [getPEIsDLL](#) ()
- static force_inline uint32_t [getPEDDataDirRVA](#) (unsigned n)
- static force_inline uint32_t [getPEDDataDirSize](#) (unsigned n)
- static force_inline uint16_t [getNumberOfSections](#) (void)
- static uint32_t [getPELFANew](#) (void)
- static force_inline int [readPESectionName](#) (unsigned char name[8], unsigned n)
- static force_inline uint32_t [getEntryPoint](#) (void)
- static force_inline uint32_t [getExeOffset](#) (void)
- static force_inline uint32_t [getImageBase](#) (void)
- static uint32_t [getVirtualEntryPoint](#) (void)
- static uint32_t [getSectionRVA](#) (unsigned i)
- static uint32_t [getSectionVirtualSize](#) (unsigned i)
- static force_inline bool [readRVA](#) (uint32_t rva, void *buf, size_t bufsize)
- static void * [memchr](#) (const void *s, int c, size_t n)
- void * [memset](#) (void *src, int c, uintptr_t n) __attribute__((nothrow)) __attribute__((__nonnull__(1)))
- void * [memmove](#) (void *dst, const void *src, uintptr_t n) __attribute__((__nothrow__)) __attribute__((__nonnull__(1)))
- void * [memcpy](#) (void *restrict dst, const void *restrict src, uintptr_t n) __attribute__((__nothrow__)) __attribute__((__nonnull__(1)))
- void * [memcmp](#) (const void *s1, const void *s2, uint32_t n) __attribute__((__nothrow__)) __attribute__((__pure__)) __attribute__((__nonnull__(1)))
- static force_inline uint32_t [DisassembleAt](#) (struct [DIS_fixed](#) *result, uint32_t offset, uint32_t len)
- static int32_t [ilog2_compat](#) (uint32_t a, uint32_t b)

16.4.1 Detailed Description

16.4.2 Define Documentation

16.4.2.1 #define BYTECODE_ABORT_HOOK 0xcea5e

entrypoint() return code that tells hook invoker that it should skip executing, probably because it'd trigger a bug in it

16.4.2.2 #define COPYRIGHT(c) const char *const _Copyright = (c);

Defines an alternative copyright for this bytecode.

config

This will also prevent the sourcecode from being embedded into the bytecode

16.4.2.3 #define DECLARE_SIGNATURE(*name*)

Value:

```
const char *name##_sig;\n    __Signature name;
```

Declares a name for a subsignature.

[config](#)

16.4.2.4 #define DEFINE_SIGNATURE(*name*, *hex*)

Value:

```
.name##_sig = (hex),\n    .name = {__COUNTER__ - __signature_bias},
```

Defines the pattern for a previously declared subsignature.

See also

[DECLARE_SIGNATURE](#)

[config](#)

Parameters

<i>name</i>	the name of a previously declared subsignature
<i>hex</i>	the pattern for this subsignature

16.4.2.5 #define FUNCTIONALITY_LEVEL_MAX(*m*) const unsigned short __FuncMax = (m);

Define the maximum engine functionality level required for this bytecode/logical signature. Engines newer than this will skip loading the bytecode. You can use the 'enum FunctionalityLevels' constants here.

[config](#)

16.4.2.6 #define FUNCTIONALITY_LEVEL_MIN(*m*) const unsigned short __FuncMin = (m);

Define the minimum engine functionality level required for this bytecode/logical signature. Engines older than this will skip loading the bytecode. You can use the 'enum FunctionalityLevels' constants here.

config

16.4.2.7 `#define ICONGROUP1(group) const char *const __IconGroup1 = (group);`

Define IconGroup1 for logical signature.

See logical signature documentation for what it is

config

16.4.2.8 `#define ICONGROUP2(group) const char *const __IconGroup2 = (group);`

Define IconGroup2 for logical signature. See logical signature documentation for what it is.

config

16.4.2.9 `#define PDF_HOOK_DECLARE const uint16_t __clambc_kind = BC_PDF;`

Make the current bytecode a PDF hook. Having a logical signature doesn't make sense here, since logical signature is evaluated AFTER these hooks run.

config

This hook is called several times, use `pdf_get_phase()` to find out in which phase you got called.

16.4.2.10 `#define PE_HOOK_DECLARE const uint16_t __clambc_kind = BC_PE_ALL;`

Make the current bytecode a PE hook, i.e. it will be called once the logical signature trigger matches (or always if there is none), and you have access to all the PE information. By default you only have access to `execs.h` information, and not to PE field information (even for PE files).

config

16.4.2.11 `#define PE_UNPACKER_DECLARE const uint16_t __clambc_kind = BC_PE_UNPACKER;`

Like `PE_HOOK_DECLARE`, but it is not run for packed files that `pe.c` can unpack (only on the unpacked file).

config

16.4.2.12 `#define SIGNATURES_DECL_BEGIN struct __Signatures {`

Marks the beginning of the subsignature name declaration section.

[config](#)

16.4.2.13 `#define SIGNATURES_DECL_END };`

Marks the end of the subsignature name declaration section.

[config](#)

16.4.2.14 `#define SIGNATURES_DEF_BEGIN`

Value:

```
static const unsigned __signature_bias = __COUNTER__+1;\nconst struct __Signatures Signatures = {\
```

Marks the beginning of subsignature pattern definitions.

[config](#)

See also

[SIGNATURES_DECL_BEGIN](#)

16.4.2.15 `#define SIGNATURES_END };\`

Marks the end of the subsignature pattern definitions.

[config](#)

16.4.2.16 `#define TARGET(tgt) const unsigned short __Target = (tgt);`

Defines the ClamAV file target.

[config](#)

Parameters

<i>tgt</i> ClamAV signature type (0 - raw, 1 - PE, etc.)
--

```
16.4.2.17 #define VIRUSNAME_PREFIX( name ) const char __clambc_virusname_prefix[] =
          name;
```

Declares the virusname prefix.

[config](#)

Parameters

<i>name</i> the prefix common to all viruses reported by this bytecode
--

```
16.4.2.18 #define VIRUSNAMES( ... ) const char *const __clambc_virusnames[] =
          {__VA_ARGS__};
```

Declares all the virusnames that this bytecode can report.

[config](#)

Parameters

<i>...</i> a comma-separated list of strings interpreted as virusnames
--

16.4.3 Function Documentation

```
16.4.3.1 bool __is_bigendian ( void ) const
```

Returns true if the bytecode is executing on a big-endian CPU.

Returns

true if executing on bigendian CPU, false otherwise

Environment

This will be optimized away in libclamav, but it must be used when dealing with endianness for portability reasons. For example whenever you read a 32-bit integer from a file, it can be written in little-endian convention (x86 CPU for example), or big-endian convention (PowerPC CPU for example). If the file always contains little-endian integers, then conversion might be needed. ClamAV bytecodes by their nature must only handle known-endian integers, if endianness can change, then both situations must be taken into account (based on a 1-byte field for example).

```
16.4.3.2 static uint16_t force_inline cli_readint16 ( const void * buff ) [static]
```

Reads from the specified buffer a 16-bit of little-endian integer.

Data structure

Parameters

in	buff	pointer to buffer
----	------	-------------------

Returns

16-bit little-endian integer converted to host endianness

16.4.3.3 `static uint32_t force_inline cli_readint32 (const void * buff) [static]`

Reads from the specified buffer a 32-bit of little-endian integer.

Data structure

Parameters

in	buff	pointer to buffer
----	------	-------------------

Returns

32-bit little-endian integer converted to host endianness

16.4.3.4 `static void force_inline cli_writeint32 (void * offset, uint32_t v) [static]`

Writes the specified value into the specified buffer in little-endian order

Data structure

Parameters

out	offset	pointer to buffer to write to
in	v	value to write

16.4.3.5 `static force_inline uint32_t count_match (__Signature sig) [static]`

Returns how many times the specified signature matched.

Parameters

sig	name of subsignature queried
-----	------------------------------

Returns

number of times this subsignature matched in the entire file

Engine query

This is a constant-time operation, the counts for all subsignatures are already computed.

16.4.3.6 void debug (...)

debug is an overloaded function (yes clang supports that in C!), but it only works on strings, and integers. Give an error on any other type

16.4.3.7 static force_inline void overloadable_func debug (const char * *str*) [static]

Prints *str* to clamscan's --debug output.

Parameters

<i>str</i>	null terminated string
------------	------------------------

16.4.3.8 static force_inline void overloadable_func debug (const uint8_t * *str*) [static]

Prints *str* to clamscan's --debug output.

Parameters

<i>str</i>	null terminated string
------------	------------------------

16.4.3.9 static force_inline void overloadable_func debug (uint32_t *a*) [static]

Prints *a* integer to clamscan's --debug output.

Parameters

<i>a</i>	integer
----------	---------

16.4.3.10 static force_inline uint32_t DisassembleAt (struct DIS_fixed * *result*, uint32_t *offset*, uint32_t *len*) [static]

Disassembles one X86 instruction starting at the specified offset.

Disassemble

Parameters

out	<i>result</i>	disassembly result
in	<i>offset</i>	start disassembling from this offset, in the current file
in	<i>len</i>	max amount of bytes to disassemble

Returns

offset where disassembly ended

16.4.3.11 `static force_inline overloadable_func void foundVirus (const char * virusname)`
`[static]`

Sets the specified virusname as the virus detected by this bytecode.

Scan**Parameters**

<i>virusname</i>	the name of the virus, excluding the prefix, must be one of the virusnames declared in VIRUSNAMES.
------------------	--

See also

[VIRUSNAMES](#)

16.4.3.12 `static force_inline void overloadable_func foundVirus (void)` `[static]`

Like [foundVirus\(\)](#) but just use the prefix as virusname

16.4.3.13 `static force_inline uint32_t getEntryPoint (void)` `[static]`

Returns the offset of the EntryPoint in the executable file.

PE**Returns**

offset of EP as 32-bit unsigned integer

16.4.3.14 `static force_inline uint32_t getExeOffset (void)` `[static]`

Returns the offset of the executable in the file.

PE**Returns**

offset of embedded executable inside file.

16.4.3.15 `static force_inline uint32_t getFileSize (void)` `[static]`

Returns the currently scanned file's size.

File operation

Returns

file size as 32-bit unsigned integer

16.4.3.16 `static force_inline uint32_t getImageBase (void) [static]`

Returns the ImageBase with the correct endian conversion. Only works if the bytecode is a PE hook (i.e. you invoked PE_UNPACKER_DECLARE)

PE

Returns

ImageBase of PE file, 0 - for non-PE hook

16.4.3.17 `static force_inline uint16_t getNumberOfSections (void) [static]`

Returns the number of sections in this executable file.

PE

Returns

number of sections as 16-bit unsigned integer

16.4.3.18 `static force_inline uint32_t getPEBaseOfCode (void) [static]`

Return the PE BaseOfCode.

PE

Returns

PE BaseOfCode, or 0 if not in PE hook.

16.4.3.19 `static force_inline uint32_t getPEBaseOfData (void) [static]`

Return the PE BaseOfData.

PE

Returns

PE BaseOfData, or 0 if not in PE hook.

16.4.3.20 `static force_inline uint16_t getPECharacteristics () [static]`

Returns PE characteristics. For example you can use this to check whether it is a DLL (0x2000).

PE

Returns

characteristic of PE file, or 0 if not in PE hook

16.4.3.21 `static force_inline uint32_t getPEChecksum (void) [static]`

Return the PE CheckSum.

PE

Returns

PE CheckSum, or 0 if not in PE hook

16.4.3.22 `static force_inline uint32_t getPEDataDirRVA (unsigned n) [static]`

Gets the virtual address of specified image data directory.

PE

Parameters

<i>n</i> image directory requested

Returns

Virtual Address of requested image directory

16.4.3.23 `static force_inline uint32_t getPEDataDirSize (unsigned n) [static]`

Gets the size of the specified image data directory.

PE

Parameters

<i>n</i> image directory requested

Returns

Size of requested image directory

16.4.3.24 `static force_inline uint16_t getPDICharacteristics (void) [static]`

Return the PE DICharacteristics.

PE

Returns

PE DICharacteristics, or 0 if not in PE hook

16.4.3.25 `static force_inline uint32_t getPEFileAlignment (void) [static]`

Return the PE FileAlignment.

PE

Returns

PE FileAlignment, or 0 if not in PE hook

16.4.3.26 `static force_inline uint64_t getPEImageBase (void) [static]`

Return the PE ImageBase as 64-bit integer.

PE

Returns

PE ImageBase as 64-bit int, or 0 if not in PE hook

16.4.3.27 `static force_inline bool getPEisDLL () [static]`

Returns whether this is a DLL. Use this only in a PE hook!

PE

Returns

true - the file is a DLL false - file is not a DLL

16.4.3.28 `static uint32_t getPELFANew (void) [static]`

Gets the offset to the PE header.

PE

Returns

offset to the PE header, or 0 if not in PE hook

16.4.3.29 `static force_inline uint32_t getPELoaderFlags (void) [static]`

Return the PE LoaderFlags.

PE

Returns

PE LoaderFlags or 0 if not in PE hook

16.4.3.30 `static force_inline uint16_t getPEMachine () [static]`

Returns the CPU this executable runs on, see libclamav/pe.c for possible values.

PE

Returns

PE Machine or 0 if not in PE hook

16.4.3.31 `static force_inline uint16_t getPEMajorImageVersion (void) [static]`

Return the PE MajorImageVersion.

PE

Returns

PE MajorImageVersion, or 0 if not in PE hook

16.4.3.32 `static static force_inline force_inline uint8_t getPEMajorLinkerVersion (void) [static]`

Returns MajorLinkerVersion for this PE file.

PE

Returns

PE MajorLinkerVersion or 0 if not in PE hook

16.4.3.33 `static force_inline uint16_t getPEMajorOperatingSystemVersion (void)`
`[static]`

Return the PE MajorOperatingSystemVersion.

PE

Returns

PE MajorOperatingSystemVersion, or 0 if not in PE hook

16.4.3.34 `static force_inline uint16_t getPEMajorSubsystemVersion (void)` `[static]`

Return the PE MajorSubsystemVersion.

PE

Returns

PE MajorSubsystemVersion or 0 if not in PE hook

16.4.3.35 `static force_inline uint16_t getPEMinorImageVersion (void)` `[static]`

Return the PE MinorImageVersion.

PE

Returns

PE MinorImageVersion, or 0 if not in PE hook

16.4.3.36 `static force_inline uint8_t getPEMinorLinkerVersion (void)` `[static]`

Returns MinorLinkerVersion for this PE file.

PE

Returns

PE MinorLinkerVersion or 0 if not in PE hook

16.4.3.37 `static force_inline uint16_t getPEMinorOperatingSystemVersion (void)`
[static]

Return the PE MinorOperatingSystemVersion.

PE

Returns

PE MinorOperatingSystemVersion, or 0 if not in PE hook

16.4.3.38 `static force_inline uint16_t getPEMinorSubsystemVersion (void)` [static]

Return the PE MinorSubsystemVersion.

PE

Returns

PE MinorSubsystemVersion, or 0 if not in PE hook

16.4.3.39 `static force_inline uint32_t getPENumberOfSymbols ()` [static]

Returns the PE number of debug symbols

PE

Returns

PE NumberOfSymbols or 0 if not in PE hook

16.4.3.40 `static force_inline uint32_t getPEPointerToSymbolTable ()` [static]

Returns pointer to the PE debug symbol table

PE

Returns

PE PointerToSymbolTable or 0 if not in PE hook

16.4.3.41 `static force_inline uint32_t getPESectionAlignment (void)` [static]

Return the PE SectionAlignment.

PE

Returns

PE SectionAlignment, or 0 if not in PE hook

16.4.3.42 `static force_inline uint32_t getPESizeOfCode (void) [static]`

Return the PE SizeOfCode.

PE

Returns

PE SizeOfCode or 0 if not in PE hook

16.4.3.43 `static force_inline uint32_t getPESizeOfHeaders (void) [static]`

Return the PE SizeOfHeaders.

PE

Returns

PE SizeOfHeaders, or 0 if not in PE hook

16.4.3.44 `static force_inline uint32_t getPESizeOfHeapCommit (void) [static]`

Return the PE SizeOfHeapCommit.

PE

Returns

PE SizeOfHeapCommit, or 0 if not in PE hook

16.4.3.45 `static force_inline uint32_t getPESizeOfHeapReserve (void) [static]`

Return the PE SizeOfHeapReserve.

PE

Returns

PE SizeOfHeapReserve, or 0 if not in PE hook

16.4.3.46 `static force_inline uint32_t getPESizeOfImage (void) [static]`

Return the PE SizeOfImage.

PE

Returns

PE SizeOfImage, or 0 if not in PE hook

16.4.3.47 `static force_inline uint32_t getPESizeOfInitializedData (void) [static]`

Return the PE SizeOfInitializedData.

PE

Returns

PE SizeOfInitializeData or 0 if not in PE hook

16.4.3.48 `static force_inline uint16_t getPESizeOfOptionalHeader () [static]`

Returns the size of PE optional header.

PE

Returns

size of PE optional header, or 0 if not in PE hook

16.4.3.49 `static force_inline uint32_t getPESizeOfStackCommit (void) [static]`

Return the PE SizeOfStackCommit.

PE

Returns

PE SizeOfStackCommit, or 0 if not in PE hook

16.4.3.50 `static force_inline uint32_t getPESizeOfStackReserve (void) [static]`

Return the PE SizeOfStackReserve.

PE

Returns

PE SizeOfStackReserver, or 0 if not in PE hook

16.4.3.51 `static force_inline uint32_t getPESizeOfUninitializedData (void) [static]`

Return the PE SizeofUninitializedData.

PE

Returns

PE SizeofUninitializedData or 0 if not in PE hook

16.4.3.52 `static force_inline uint16_t getPESubsystem (void) [static]`

Return the PE Subsystem.

PE

Returns

PE subsystem, or 0 if not in PE hook

16.4.3.53 `static force_inline uint32_t getPETimeDateStamp () [static]`

Returns the PE TimeDateStamp from headers

PE

Returns

PE TimeDateStamp or 0 if not in PE hook

16.4.3.54 `static force_inline uint32_t getPEWin32VersionValue (void) [static]`

Return the PE Win32VersionValue.

PE

Returns

PE Win32VersionValue, or 0 if not in PE hook

16.4.3.55 `static uint32_t getSectionRVA (unsigned i) [static]`

Return the RVA of the specified section

PE

Parameters

<i>i</i> section index (from 0)

Returns

RVA of section, or -1 if invalid

16.4.3.56 `static uint32_t getSectionVirtualSize (unsigned i) [static]`

Return the virtual size of the specified section.

PE

Parameters

<i>i</i> section index (from 0)

Returns

VSZ of section, or -1 if invalid

16.4.3.57 `static uint32_t getVirtualEntryPoint (void) [static]`

The address of the EntryPoint. Use this for matching EP against sections.

PE

Returns

virtual address of EntryPoint, or 0 if not in PE hook

16.4.3.58 `static force_inline bool hasExeInfo (void) [static]`

Returns whether the current file has executable information.

PE

Returns

true if the file has exe info, false otherwise

16.4.3.59 `static force_inline bool hasPEInfo (void) [static]`

Returns whether PE information is available

PE

Returns

true if PE information is available (in PE hooks)

16.4.3.60 `static int32_t ilog2_compat (uint32_t a, uint32_t b) [inline, static]`

ilog2_compat for 0.96 compatibility, you should use [ilog2\(\)](#) 0.96.1 API instead of this one!

16.4.3.61 `static force_inline bool isPE64 (void) [static]`

Returns whether this is a PE32+ executable.

PE

Returns

true if this is a PE32+ executable

16.4.3.62 `static uint16_t force_inline le16_to_host (uint16_t v) [static]`

Converts the specified value if needed, knowing it is in little endian order.

Data structure

Parameters

<code>in</code>	<code>v</code>	16-bit integer as read from a file
-----------------	----------------	------------------------------------

Returns

integer converted to host's endianness

16.4.3.63 `static uint32_t force_inline le32_to_host (uint32_t v) [static]`

Converts the specified value if needed, knowing it is in little endian order.

Data structure

Parameters

<code>in</code>	<code>v</code>	32-bit integer as read from a file
-----------------	----------------	------------------------------------

Returns

integer converted to host's endianness

16.4.3.64 `static uint64_t force_inline le64_to_host (uint64_t v) [static]`

Converts the specified value if needed, knowing it is in little endian order.

Data structure**Parameters**

<code>in</code>	<code>v</code>	64-bit integer as read from a file
-----------------	----------------	------------------------------------

Returns

integer converted to host's endianness

16.4.3.65 `static force_inline uint32_t match_location (__Signature sig, uint32_t goback) [static]`

Returns the offset of the match.

Engine query**Parameters**

<code>sig</code>	- Signature
<code>goback</code>	- max length of signature

Returns

offset of match

16.4.3.66 `static force_inline int32_t match_location_check (__Signature sig, uint32_t goback, const char * static_start, uint32_t static_len) [static]`

Like [match_location\(\)](#), but also checks that the match starts with the specified hex string.

Engine query

It is recommended to use this for safety and compatibility with 0.96.1

Parameters

<i>sig</i>	- signature
<i>goback</i>	- maximum length of signature (till start of last subsig)
<i>static_start</i>	- static string that sig must begin with
<i>static_len</i>	- static string that sig must begin with - length

Returns

>=0 - offset of match -1 - no match

16.4.3.67 `static force_inline uint32_t matches (__Signature sig) [static]`

Returns whether the specified subsignature has matched at least once.

Engine query**Parameters**

<i>sig</i>	name of subsignature queried
------------	------------------------------

Returns

1 if subsignature one or more times, 0 otherwise

16.4.3.68 `static void* memchr (const void * s, int c, size_t n) [static]`

Scan the first *n* bytes of the buffer *s*, for the character *c*.

String operation**Parameters**

<i>in</i>	<i>s</i>	buffer to scan
	<i>c</i>	character to look for
	<i>n</i>	size of buffer

Returns

a pointer to the first byte to match, or NULL if not found.

16.4.3.69 `void* void int memcmp (const void * s1, const void * s2, uint32_t n)`

Compares two memory buffers.

String operation

Parameters

in	<i>s1</i>	buffer one
in	<i>s2</i>	buffer two
in	<i>n</i>	amount of bytes to copy

Returns

an integer less than, equal to, or greater than zero if the first *n* bytes of *s1* are found, respectively, to be less than, to match, or be greater than the first *n* bytes of *s2*.

16.4.3.70 void* void memcpy (void *restrict *dst*, const void *restrict *src*, uintptr_t *n*)

Copies data between two non-overlapping buffers.

String operation**Parameters**

out	<i>dst</i>	destination buffer
in	<i>src</i>	source buffer
in	<i>n</i>	amount of bytes to copy

Returns

dst

16.4.3.71 void* memmove (void * *dst*, const void * *src*, uintptr_t *n*)

Copies data between two possibly overlapping buffers.

String operation**Parameters**

out	<i>dst</i>	destination buffer
in	<i>src</i>	source buffer
in	<i>n</i>	amount of bytes to copy

Returns

dst

16.4.3.72 void* memset (void * *src*, int *c*, uintptr_t *n*)

Fills the specified buffer to the specified value.

String operation

Parameters

out	<i>src</i>	pointer to buffer
in	<i>c</i>	character to fill buffer with
in	<i>n</i>	length of buffer

Returns

src

16.4.3.73 `static force_inline int readPESectionName (unsigned char name[8], unsigned n)`
`[static]`

Read name of requested PE section.

PE

Parameters

out	<i>name</i>	name of PE section
in	<i>n</i>	PE section requested

Returns

0 if successful, <0 otherwise

16.4.3.74 `static force_inline bool readRVA (uint32_t rva, void * buf, size_t bufsize)`
`[static]`

read the specified amount of bytes from the PE file, starting at the address specified by RVA.

PE

Parameters

	<i>rva</i>	the Relative Virtual Address you want to read from (will be converted to file offset)
out	<i>buf</i>	destination buffer
	<i>bufsize</i>	size of buffer

Returns

true on success (full read), false on any failure

16.5 bytecode_pe.h File Reference

Data Structures

- struct [pe_image_file_hdr](#)
- struct [pe_image_data_dir](#)
- struct [pe_image_optional_hdr32](#)
- struct [pe_image_optional_hdr64](#)
- struct [pe_image_section_hdr](#)
- struct [cli_pe_hook_data](#)

16.5.1 Detailed Description

Index

- __clambc_filesize
bytecode_api.h, [53](#)
 - __clambc_kind
bytecode_api.h, [53](#)
 - __clambc_match_counts
bytecode_api.h, [53](#)
 - __clambc_match_offsets
bytecode_api.h, [53](#)
 - __clambc_pedata
bytecode_api.h, [53](#)
 - __is_bigendian
bytecode_local.h, [70](#)
- ACCESS_IMM
bytecode_disasm.h, [56](#)
- ACCESS_MEM
bytecode_disasm.h, [56](#)
- ACCESS_NOARG
bytecode_disasm.h, [56](#)
- ACCESS_REG
bytecode_disasm.h, [56](#)
- ACCESS_REL
bytecode_disasm.h, [56](#)
- access_size
DIS_arg, [16](#)
DIS_mem_arg, [18](#)
- access_type
DIS_arg, [16](#)
- add_reg
DIS_mem_arg, [18](#)
- address_size
DIS_fixed, [17](#)
- atoi
bytecode_api.h, [27](#)
- BC_GENERIC
bytecode_api.h, [27](#)
- BC_LOGICAL
bytecode_api.h, [27](#)
- BC_PE_UNPACKER
bytecode_api.h, [27](#)
- buffer_pipe_done
bytecode_api.h, [28](#)
- buffer_pipe_new
bytecode_api.h, [28](#)
- buffer_pipe_new_fromfile
bytecode_api.h, [28](#)
- buffer_pipe_read_avail
bytecode_api.h, [29](#)
- buffer_pipe_read_get
bytecode_api.h, [29](#)
- buffer_pipe_read_stopped
bytecode_api.h, [29](#)
- buffer_pipe_write_avail
bytecode_api.h, [29](#)
- buffer_pipe_write_get
bytecode_api.h, [30](#)
- buffer_pipe_write_stopped
bytecode_api.h, [30](#)
- bytecode_api.h
 - BC_GENERIC, [27](#)
 - BC_LOGICAL, [27](#)
 - BC_PE_UNPACKER, [27](#)
 - PE_INVALID_RVA, [26](#)
 - SEEK_CUR, [27](#)
 - SEEK_END, [27](#)
 - SEEK_SET, [27](#)
- bytecode_disasm.h
 - ACCESS_IMM, [56](#)
 - ACCESS_MEM, [56](#)
 - ACCESS_NOARG, [56](#)
 - ACCESS_REG, [56](#)
 - ACCESS_REL, [56](#)
 - OP_AAA, [56](#)
 - OP_AAD, [56](#)
 - OP_AAM, [56](#)
 - OP_AAS, [56](#)
 - OP_ADC, [57](#)
 - OP_ADD, [56](#)
 - OP_AND, [57](#)
 - OP_ARPL, [57](#)
 - OP_BOUND, [57](#)
 - OP_BSF, [57](#)
 - OP_BSR, [57](#)
 - OP_BSWAP, [57](#)
 - OP_BT, [57](#)
 - OP BTC, [57](#)
 - OP_BTR, [57](#)
 - OP BTS, [57](#)
 - OP_CALL, [57](#)
 - OP_CBW, [57](#)
 - OP_CDQ, [57](#)
 - OP_CLC, [57](#)

OP_CLD, [57](#)
OP_CLI, [57](#)
OP_CLTS, [57](#)
OP_CMC, [57](#)
OP_CMOVA, [57](#)
OP_CMOVBE, [57](#)
OP_CMOVC, [57](#)
OP_CMOVG, [57](#)
OP_CMOVGE, [57](#)
OP_CMOVL, [57](#)
OP_CMOVLE, [57](#)
OP_CMOVNC, [57](#)
OP_CMOVNO, [57](#)
OP_CMOVNP, [57](#)
OP_CMOVNS, [57](#)
OP_CMOVNZ, [57](#)
OP_CMOVO, [57](#)
OP_CMOVP, [57](#)
OP_CMOVS, [57](#)
OP_CMOVZ, [57](#)
OP_CMP, [57](#)
OP_CMPSB, [58](#)
OP_CMPSD, [57](#)
OP_CMPSW, [58](#)
OP_CMPXCHG, [58](#)
OP_CMPXCHG8B, [58](#)
OP_CPUID, [58](#)
OP_CWDE, [57](#)
OP_DAA, [58](#)
OP_DAS, [58](#)
OP_DEC, [58](#)
OP_DIV, [58](#)
OP_ENTER, [58](#)
OP_F2XM1, [61](#)
OP_FABS, [61](#)
OP_FADD, [61](#)
OP_FADDP, [62](#)
OP_FBLD, [62](#)
OP_FBSTP, [62](#)
OP_FCHS, [62](#)
OP_FCLEX, [62](#)
OP_FCMOVBE, [62](#)
OP_FCMOVBE, [62](#)
OP_FCMOVE, [62](#)
OP_FCMOVNB, [62](#)
OP_FCMOVNBE, [62](#)
OP_FCMOVNE, [62](#)
OP_FCMOVNU, [62](#)
OP_FCMOVU, [62](#)
OP_FCOM, [62](#)
OP_FCOMI, [62](#)
OP_FCOMIP, [62](#)
OP_FCOMP, [62](#)
OP_FCOMPP, [62](#)
OP_FCOS, [62](#)
OP_FDECSTP, [62](#)
OP_FDIV, [62](#)
OP_FDIVP, [62](#)
OP_FDIVR, [62](#)
OP_FDIVRP, [62](#)
OP_FFREE, [62](#)
OP_FIADD, [62](#)
OP_FICOM, [62](#)
OP_FICOMP, [62](#)
OP_FIDIV, [62](#)
OP_FIDIVR, [62](#)
OP_FILD, [62](#)
OP_FIMUL, [62](#)
OP_FINCSTP, [62](#)
OP_FINIT, [62](#)
OP_FIST, [62](#)
OP_FISTP, [62](#)
OP_FISTTP, [62](#)
OP_FISUB, [62](#)
OP_FISUBR, [63](#)
OP_FLD, [63](#)
OP_FLD1, [63](#)
OP_FLDCW, [63](#)
OP_FLDENV, [63](#)
OP_FLDL2E, [63](#)
OP_FLDL2T, [63](#)
OP_FLDLG2, [63](#)
OP_FLDLN2, [63](#)
OP_FLDPI, [63](#)
OP_FLDZ, [63](#)
OP_FMUL, [63](#)
OP_FMULP, [63](#)
OP_FNOP, [63](#)
OP_FPATAN, [63](#)
OP_FPREM, [63](#)
OP_FPREM1, [63](#)
OP_FPTAN, [63](#)
OP_FPU, [61](#)
OP_FRNDINT, [63](#)
OP_FRSTOR, [63](#)
OP_FSAVE, [63](#)
OP_FSCALE, [63](#)
OP_FSINCOS, [63](#)
OP_FSQRT, [63](#)
OP_FST, [63](#)

OP_FSTCW, 63	OP_JS, 58
OP_FSTENV, 63	OP_JZ, 58
OP_FSTP, 63	OP_LAHF, 59
OP_FSTSW, 63	OP_LAR, 59
OP_FSUB, 63	OP_LDS, 59
OP_FSUBP, 63	OP_LEA, 59
OP_FSUBR, 63	OP_LEAVE, 59
OP_FSUBRP, 63	OP_LES, 59
OP_FTST, 63	OP_LFS, 59
OP_FUCOM, 63	OP_LGDT, 59
OP_FUCOMI, 63	OP_LGS, 59
OP_FUCOMIP, 63	OP_LIDT, 59
OP_FUCOMP, 63	OP_LLDT, 59
OP_FUCOMPP, 64	OP_LODSB, 59
OP_FWAIT, 58	OP_LODSD, 59
OP_FXAM, 64	OP_LODSW, 59
OP_FXCH, 64	OP_LOOP, 59
OP_FXTRACT, 64	OP_LOOPE, 59
OP_FYL2X, 64	OP_LOOPNE, 59
OP_FYL2XP1, 64	OP_LSL, 59
OP_HLT, 58	OP_LSS, 59
OP_IDIV, 58	OP_LTR, 59
OP_IMUL, 58	OP_MOV, 59
OP_IN, 58	OP_MOVSB, 59
OP_INC, 58	OP_MOVSD, 59
OP_INSB, 58	OP_MOVSW, 59
OP_INSD, 58	OP_MOVSX, 59
OP_INSW, 58	OP_MOVZX, 59
OP_INT, 58	OP_MUL, 59
OP_INT3, 58	OP_NEG, 59
OP_INT0, 58	OP_NOP, 59
OP_INV, 58	OP_NOT, 59
OP_INVLPG, 58	OP_OR, 59
OP_IRET, 58	OP_OUT, 59
OP_JA, 58	OP_OUTSB, 60
OP_JBE, 58	OP_OUTSD, 60
OP_JC, 58	OP_OUTSW, 60
OP_JECXZ, 59	OP_POP, 60
OP_JG, 59	OP_POPAD, 60
OP_JGE, 59	OP_POPFD, 60
OP_JL, 58	OP_PREFIX_LOCK, 59
OP_JLE, 59	OP_PREFIX_REPE, 60
OP_JMP, 59	OP_PREFIX_REPN, 60
OP_JNC, 58	OP_PUSH, 60
OP_JNO, 58	OP_PUSHAD, 60
OP_JNP, 58	OP_PUSHFD, 60
OP_JNS, 58	OP_RCL, 60
OP_JNZ, 58	OP_RCR, 60
OP_JO, 58	OP_RDMSR, 60
OP_JP, 58	OP_RDPMSR, 60

OP_RDTSC, [60](#)
OP_RETF, [60](#)
OP_RETN, [60](#)
OP_ROL, [60](#)
OP_ROR, [60](#)
OP_RSM, [60](#)
OP_SAHF, [60](#)
OP_SAR, [60](#)
OP_SBB, [60](#)
OP_SCASB, [60](#)
OP_SCASD, [60](#)
OP_SCASW, [60](#)
OP_SETA, [60](#)
OP_SETBE, [60](#)
OP_SETC, [60](#)
OP_SETG, [61](#)
OP_SETGE, [61](#)
OP_SETL, [61](#)
OP_SETLE, [61](#)
OP_SETNC, [60](#)
OP_SETNO, [60](#)
OP_SETNP, [61](#)
OP_SETNS, [60](#)
OP_SETNZ, [60](#)
OP_SETO, [60](#)
OP_SETP, [60](#)
OP_SETS, [60](#)
OP_SETZ, [60](#)
OP_SGDT, [61](#)
OP_SHL, [61](#)
OP_SHLD, [61](#)
OP_SHR, [61](#)
OP_SHRD, [61](#)
OP_SIDT, [61](#)
OP_SLDT, [61](#)
OP_STC, [61](#)
OP_STD, [61](#)
OP_STI, [61](#)
OP_STOSB, [61](#)
OP_STOSD, [61](#)
OP_STOSW, [61](#)
OP_STR, [61](#)
OP_SUB, [61](#)
OP_SYSCALL, [61](#)
OP_SYSENTER, [61](#)
OP_SYSEXIT, [61](#)
OP_SYSRET, [61](#)
OP_TEST, [61](#)
OP_UD2, [61](#)
OP_VERR, [61](#)
OP_VERRW, [61](#)
OP_WBINVD, [61](#)
OP_WRMSR, [61](#)
OP_XADD, [61](#)
OP_XCHG, [61](#)
OP_XLAT, [61](#)
OP_XOR, [61](#)
SIZEB, [56](#)
SIZED, [56](#)
SIZEF, [56](#)
SIZEPTR, [56](#)
SIZEQ, [56](#)
SIZET, [56](#)
SIZEW, [56](#)
BYTECODE_ABORT_HOOK
 bytecode_local.h, [67](#)
bytecode_api.h, [24](#)
 __clambc_filesize, [53](#)
 __clambc_kind, [53](#)
 __clambc_match_counts, [53](#)
 __clambc_match_offsets, [53](#)
 __clambc_pedata, [53](#)
 atoi, [27](#)
 buffer_pipe_done, [28](#)
 buffer_pipe_new, [28](#)
 buffer_pipe_new_fromfile, [28](#)
 buffer_pipe_read_avail, [29](#)
 buffer_pipe_read_get, [29](#)
 buffer_pipe_read_stopped, [29](#)
 buffer_pipe_write_avail, [29](#)
 buffer_pipe_write_get, [30](#)
 buffer_pipe_write_stopped, [30](#)
 bytecode_rt_error, [30](#)
 BytecodeKind, [27](#)
 check_platform, [31](#)
 debug_print_str, [31](#)
 debug_print_str_nonl, [31](#)
 debug_print_str_start, [32](#)
 debug_print_uint, [32](#)
 disable_bytecode_if, [32](#)
 disable_jit_if, [33](#)
 disasm_x86, [33](#)
 engine_db_options, [33](#)
 engine_dconf_level, [34](#)
 engine_functionality_level, [34](#)
 engine_scan_options, [34](#)
 entropy_buffer, [34](#)
 extract_new, [35](#)
 extract_set_container, [35](#)
 file_byteat, [35](#)

file_find, 36
file_find_limit, 36
fill_buffer, 36
FunctionalityLevels, 27
get_environment, 37
get_pe_section, 37
hashset_add, 37
hashset_contains, 38
hashset_done, 38
hashset_empty, 38
hashset_new, 39
hashset_remove, 39
hex2ui, 39
icos, 40
iexp, 40
ilog2, 40
inflate_done, 41
inflate_init, 41
inflate_process, 41
input_switch, 42
ipow, 42
isin, 42
jsnorm_done, 43
jsnorm_init, 43
jsnorm_process, 43
malloc, 44
map_addkey, 44
map_done, 44
map_find, 45
map_getvalue, 45
map_getvaluesize, 45
map_new, 46
map_remove, 46
map_setvalue, 46
matchicon, 47
memstr, 47
pdf_flag, 27
pdf_get_dumpedobjid, 47
pdf_get_flags, 48
pdf_get_obj_num, 48
pdf_get_phase, 48
pdf_getobj, 48
pdf_getobjsize, 49
pdf_lookupobj, 49
pdf_objflags, 27
pdf_phase, 27
pdf_set_flags, 49
pe_rawaddr, 50
read, 50
read_number, 50
seek, 51
setvirusname, 51
test1, 51
test2, 52
version_compare, 52
write, 52
bytecode_disasm.h, 53
DIS_ACCESS, 56
DIS_SIZE, 56
X86OPS, 56
X86REGS, 64
bytecode_execs.h, 64
bytecode_local.h, 64
__is_bigendian, 70
BYTECODE_ABORT_HOOK, 67
cli_readint16, 71
cli_readint32, 71
cli_writeint32, 71
COPYRIGHT, 67
count_match, 72
debug, 72
DECLARE_SIGNATURE, 67
DEFINE_SIGNATURE, 67
DisassembleAt, 72
foundVirus, 73
FUNCTIONALITY_LEVEL_MAX, 67
FUNCTIONALITY_LEVEL_MIN, 68
getEntryPoint, 73
getExeOffset, 73
getFileSize, 74
getImageBase, 74
getNumberOfSections, 74
getPEBaseOfCode, 74
getPEBaseOfData, 75
getPECharacteristics, 75
getPEChecksum, 75
getPEDataDirRVA, 75
getPEDataDirSize, 76
getPEDllCharacteristics, 76
getPEFileAlignment, 76
getPEImageBase, 76
getPEIsDLL, 77
getPELFANew, 77
getPELoaderFlags, 77
getPEMachine, 77
getPEMajorImageVersion, 77
getPEMajorLinkerVersion, 78
getPEMajorOperatingSystemVersion,
78
getPEMajorSubsystemVersion, 78

getPEMinorImageVersion, 78
getPEMinorLinkerVersion, 79
getPEMinorOperatingSystemVersion, 79
getPEMinorSubsystemVersion, 79
getPENumberOfSymbols, 79
getPEPointerToSymbolTable, 80
getPESectionAlignment, 80
getPESizeOfCode, 80
getPESizeOfHeaders, 80
getPESizeOfHeapCommit, 80
getPESizeOfHeapReserve, 81
getPESizeOfImage, 81
getPESizeOfInitializedData, 81
getPESizeOfOptionalHeader, 81
getPESizeOfStackCommit, 82
getPESizeOfStackReserve, 82
getPESizeOfUninitializedData, 82
getPESubsystem, 82
getPETimeDateStamp, 82
getPEWin32VersionValue, 83
getSectionRVA, 83
getSectionVirtualSize, 83
getVirtualEntryPoint, 84
hasExeInfo, 84
hasPEInfo, 84
ICONGROUP1, 68
ICONGROUP2, 68
ilog2_compat, 84
isPE64, 84
le16_to_host, 84
le32_to_host, 85
le64_to_host, 85
match_location, 85
match_location_check, 86
matches, 86
memchr, 86
memcmp, 87
memcpy, 87
memmove, 87
memset, 88
PDF_HOOK_DECLARE, 68
PE_HOOK_DECLARE, 68
PE_UNPACKER_DECLARE, 69
readPESectionName, 88
readRVA, 88
SIGNATURES_DECL_BEGIN, 69
SIGNATURES_DECL_END, 69
SIGNATURES_DEF_BEGIN, 69
SIGNATURES_END, 69
TARGET, 70
VIRUSNAME_PREFIX, 70
VIRUSNAMES, 70
bytecode_pe.h, 89
bytecode_rt_error
 bytecode_api.h, 30
BytecodeKind
 bytecode_api.h, 27
check_platform
 bytecode_api.h, 31
Checksum
 pe_image_optional_hdr32, 20
 pe_image_optional_hdr64, 22
chr
 cli_exe_section, 14
cli_exe_info, 12
 ep, 13
 hdr_size, 13
 nsections, 13
 offset, 13
 res_addr, 13
 section, 13
cli_exe_section, 13
 chr, 14
 raw, 14
 rsz, 14
 rva, 14
 uraw, 14
 ursz, 14
 urva, 14
 uvsz, 14
 vsz, 14
cli_pe_hook_data, 14
 dirs, 15
 e_lfanew, 15
 ep, 15
 file_hdr, 15
 hdr_size, 15
 nsections, 15
 opt32, 15
 opt64, 15
 overlays, 15
 overlays_sz, 16
cli_readint16
 bytecode_local.h, 71
cli_readint32
 bytecode_local.h, 71
cli_writeint32
 bytecode_local.h, 71

- COPYRIGHT
 - bytecode_local.h, 67
- count_match
 - bytecode_local.h, 72
- debug
 - bytecode_local.h, 72
- debug_print_str
 - bytecode_api.h, 31
- debug_print_str_nonl
 - bytecode_api.h, 31
- debug_print_str_start
 - bytecode_api.h, 32
- debug_print_uint
 - bytecode_api.h, 32
- DECLARE_SIGNATURE
 - bytecode_local.h, 67
- DEFINE_SIGNATURE
 - bytecode_local.h, 67
- dirs
 - cli_pe_hook_data, 15
- DIS_ACCESS
 - bytecode_disasm.h, 56
- DIS_arg, 16
 - access_size, 16
 - access_type, 16
 - mem, 16
 - other, 16
 - reg, 16
- DIS_fixed, 17
 - address_size, 17
 - operation_size, 17
 - segment, 17
 - x86_opcode, 17
- DIS_mem_arg, 17
 - access_size, 18
 - add_reg, 18
 - displacement, 18
 - scale, 18
 - scale_reg, 18
- DIS_SIZE
 - bytecode_disasm.h, 56
- disable_bytecode_if
 - bytecode_api.h, 32
- disable_jit_if
 - bytecode_api.h, 33
- DISASM_RESULT, 18
- disasm_x86
 - bytecode_api.h, 33
- DisassembleAt
 - bytecode_local.h, 72
- displacement
 - DIS_mem_arg, 18
- e_lfanew
 - cli_pe_hook_data, 15
- engine_db_options
 - bytecode_api.h, 33
- engine_dconf_level
 - bytecode_api.h, 34
- engine_functionality_level
 - bytecode_api.h, 34
- engine_scan_options
 - bytecode_api.h, 34
- entropy_buffer
 - bytecode_api.h, 34
- ep
 - cli_exe_info, 13
 - cli_pe_hook_data, 15
- extract_new
 - bytecode_api.h, 35
- extract_set_container
 - bytecode_api.h, 35
- file_byteat
 - bytecode_api.h, 35
- file_find
 - bytecode_api.h, 36
- file_find_limit
 - bytecode_api.h, 36
- file_hdr
 - cli_pe_hook_data, 15
- FileAlignment
 - pe_image_optional_hdr32, 20
 - pe_image_optional_hdr64, 22
- fill_buffer
 - bytecode_api.h, 36
- foundVirus
 - bytecode_local.h, 73
- FUNCTIONALITY_LEVEL_MAX
 - bytecode_local.h, 67
- FUNCTIONALITY_LEVEL_MIN
 - bytecode_local.h, 68
- FunctionalityLevels
 - bytecode_api.h, 27
- get_environment
 - bytecode_api.h, 37
- get_pe_section
 - bytecode_api.h, 37

- getEntryPoint
 - bytecode_local.h, [73](#)
- getExeOffset
 - bytecode_local.h, [73](#)
- getFileSize
 - bytecode_local.h, [74](#)
- getImageBase
 - bytecode_local.h, [74](#)
- getNumberOfSections
 - bytecode_local.h, [74](#)
- getPEBaseOfCode
 - bytecode_local.h, [74](#)
- getPEBaseOfData
 - bytecode_local.h, [75](#)
- getPECharacteristics
 - bytecode_local.h, [75](#)
- getPEChecksum
 - bytecode_local.h, [75](#)
- getPEDataDirRVA
 - bytecode_local.h, [75](#)
- getPEDataDirSize
 - bytecode_local.h, [76](#)
- getPEDllCharacteristics
 - bytecode_local.h, [76](#)
- getPEFileAlignment
 - bytecode_local.h, [76](#)
- getPEImageBase
 - bytecode_local.h, [76](#)
- getPEIsDLL
 - bytecode_local.h, [77](#)
- getPELFANew
 - bytecode_local.h, [77](#)
- getPELoaderFlags
 - bytecode_local.h, [77](#)
- getPEMachine
 - bytecode_local.h, [77](#)
- getPEMajorImageVersion
 - bytecode_local.h, [77](#)
- getPEMajorLinkerVersion
 - bytecode_local.h, [78](#)
- getPEMajorOperatingSystemVersion
 - bytecode_local.h, [78](#)
- getPEMajorSubsystemVersion
 - bytecode_local.h, [78](#)
- getPEMinorImageVersion
 - bytecode_local.h, [78](#)
- getPEMinorLinkerVersion
 - bytecode_local.h, [79](#)
- getPEMinorOperatingSystemVersion
 - bytecode_local.h, [79](#)
- getPEMinorSubsystemVersion
 - bytecode_local.h, [79](#)
- getPENumberOfSymbols
 - bytecode_local.h, [79](#)
- getPEPointerToSymbolTable
 - bytecode_local.h, [80](#)
- getPESectionAlignment
 - bytecode_local.h, [80](#)
- getPESizeOfCode
 - bytecode_local.h, [80](#)
- getPESizeOfHeaders
 - bytecode_local.h, [80](#)
- getPESizeOfHeapCommit
 - bytecode_local.h, [80](#)
- getPESizeOfHeapReserve
 - bytecode_local.h, [81](#)
- getPESizeOfImage
 - bytecode_local.h, [81](#)
- getPESizeOfInitializedData
 - bytecode_local.h, [81](#)
- getPESizeOfOptionalHeader
 - bytecode_local.h, [81](#)
- getPESizeOfStackCommit
 - bytecode_local.h, [82](#)
- getPESizeOfStackReserve
 - bytecode_local.h, [82](#)
- getPESizeOfUninitializedData
 - bytecode_local.h, [82](#)
- getPESubsystem
 - bytecode_local.h, [82](#)
- getPETimeDateStamp
 - bytecode_local.h, [82](#)
- getPEWin32VersionValue
 - bytecode_local.h, [83](#)
- getSectionRVA
 - bytecode_local.h, [83](#)
- getSectionVirtualSize
 - bytecode_local.h, [83](#)
- getVirtualEntryPoint
 - bytecode_local.h, [84](#)
- hasExeInfo
 - bytecode_local.h, [84](#)
- hashset_add
 - bytecode_api.h, [37](#)
- hashset_contains
 - bytecode_api.h, [38](#)
- hashset_done
 - bytecode_api.h, [38](#)
- hashset_empty

- bytecode_api.h, [38](#)
- hashset_new
 - bytecode_api.h, [39](#)
- hashset_remove
 - bytecode_api.h, [39](#)
- hasPEInfo
 - bytecode_local.h, [84](#)
- hdr_size
 - cli_exe_info, [13](#)
 - cli_pe_hook_data, [15](#)
- hex2ui
 - bytecode_api.h, [39](#)
- ICONGROUP1
 - bytecode_local.h, [68](#)
- ICONGROUP2
 - bytecode_local.h, [68](#)
- icos
 - bytecode_api.h, [40](#)
- iexp
 - bytecode_api.h, [40](#)
- ilog2
 - bytecode_api.h, [40](#)
- ilog2_compat
 - bytecode_local.h, [84](#)
- ImageBase
 - pe_image_optional_hdr32, [20](#)
 - pe_image_optional_hdr64, [22](#)
- inflate_done
 - bytecode_api.h, [41](#)
- inflate_init
 - bytecode_api.h, [41](#)
- inflate_process
 - bytecode_api.h, [41](#)
- input_switch
 - bytecode_api.h, [42](#)
- ipow
 - bytecode_api.h, [42](#)
- isin
 - bytecode_api.h, [42](#)
- isPE64
 - bytecode_local.h, [84](#)
- jsnorm_done
 - bytecode_api.h, [43](#)
- jsnorm_init
 - bytecode_api.h, [43](#)
- jsnorm_process
 - bytecode_api.h, [43](#)
- le16_to_host
 - bytecode_local.h, [84](#)
- le32_to_host
 - bytecode_local.h, [85](#)
- le64_to_host
 - bytecode_local.h, [85](#)
- Machine
 - pe_image_file_hdr, [19](#)
- Magic
 - pe_image_file_hdr, [19](#)
- MajorImageVersion
 - pe_image_optional_hdr32, [20](#)
 - pe_image_optional_hdr64, [22](#)
- MajorLinkerVersion
 - pe_image_optional_hdr32, [20](#)
 - pe_image_optional_hdr64, [22](#)
- MajorOperatingSystemVersion
 - pe_image_optional_hdr32, [20](#)
 - pe_image_optional_hdr64, [22](#)
- malloc
 - bytecode_api.h, [44](#)
- map_addkey
 - bytecode_api.h, [44](#)
- map_done
 - bytecode_api.h, [44](#)
- map_find
 - bytecode_api.h, [45](#)
- map_getvalue
 - bytecode_api.h, [45](#)
- map_getvaluesize
 - bytecode_api.h, [45](#)
- map_new
 - bytecode_api.h, [46](#)
- map_remove
 - bytecode_api.h, [46](#)
- map_setvalue
 - bytecode_api.h, [46](#)
- match_location
 - bytecode_local.h, [85](#)
- match_location_check
 - bytecode_local.h, [86](#)
- matches
 - bytecode_local.h, [86](#)
- matchicon
 - bytecode_api.h, [47](#)
- mem
 - DIS_arg, [16](#)
- memchr
 - bytecode_local.h, [86](#)
- memcmp

- bytecode_local.h, [87](#)
- memcpy
 - bytecode_local.h, [87](#)
- memcpymove
 - bytecode_local.h, [87](#)
- memset
 - bytecode_local.h, [88](#)
- memstr
 - bytecode_api.h, [47](#)
- MinorImageVersion
 - pe_image_optional_hdr32, [20](#)
 - pe_image_optional_hdr64, [22](#)
- MinorLinkerVersion
 - pe_image_optional_hdr32, [21](#)
 - pe_image_optional_hdr64, [22](#)
- MinorOperatingSystemVersion
 - pe_image_optional_hdr32, [21](#)
 - pe_image_optional_hdr64, [22](#)
- Name
 - pe_image_section_hdr, [23](#)
- nsections
 - cli_exe_info, [13](#)
 - cli_pe_hook_data, [15](#)
- NumberOfLinenumbers
 - pe_image_section_hdr, [23](#)
- NumberOfRelocations
 - pe_image_section_hdr, [23](#)
- NumberOfRvaAndSizes
 - pe_image_optional_hdr32, [21](#)
 - pe_image_optional_hdr64, [22](#)
- NumberOfSections
 - pe_image_file_hdr, [19](#)
- NumberOfSymbols
 - pe_image_file_hdr, [19](#)
- offset
 - cli_exe_info, [13](#)
- OP_AAA
 - bytecode_disasm.h, [56](#)
- OP_AAD
 - bytecode_disasm.h, [56](#)
- OP_AAM
 - bytecode_disasm.h, [56](#)
- OP_AAS
 - bytecode_disasm.h, [56](#)
- OP_ADC
 - bytecode_disasm.h, [57](#)
- OP_ADD
 - bytecode_disasm.h, [56](#)
- OP_AND
 - bytecode_disasm.h, [57](#)
- OP_ARPL
 - bytecode_disasm.h, [57](#)
- OP_BOUND
 - bytecode_disasm.h, [57](#)
- OP_BSF
 - bytecode_disasm.h, [57](#)
- OP_BSR
 - bytecode_disasm.h, [57](#)
- OP_BSWAP
 - bytecode_disasm.h, [57](#)
- OP_BT
 - bytecode_disasm.h, [57](#)
- OPBTC
 - bytecode_disasm.h, [57](#)
- OP_BTR
 - bytecode_disasm.h, [57](#)
- OP_BTS
 - bytecode_disasm.h, [57](#)
- OP_CALL
 - bytecode_disasm.h, [57](#)
- OP_CBW
 - bytecode_disasm.h, [57](#)
- OP_CDQ
 - bytecode_disasm.h, [57](#)
- OP_CLC
 - bytecode_disasm.h, [57](#)
- OP_CLD
 - bytecode_disasm.h, [57](#)
- OP_CLI
 - bytecode_disasm.h, [57](#)
- OP_CLTS
 - bytecode_disasm.h, [57](#)
- OP_CMC
 - bytecode_disasm.h, [57](#)
- OP_CMOVA
 - bytecode_disasm.h, [57](#)
- OP_CMOVBE
 - bytecode_disasm.h, [57](#)
- OP_CMOVC
 - bytecode_disasm.h, [57](#)
- OP_CMOVG
 - bytecode_disasm.h, [57](#)
- OP_CMOVGE
 - bytecode_disasm.h, [57](#)
- OP_CMOVL
 - bytecode_disasm.h, [57](#)
- OP_CMOVLE
 - bytecode_disasm.h, [57](#)

- OP_CMOVNC
 - [bytecode_disasm.h, 57](#)
- OP_CMOVNO
 - [bytecode_disasm.h, 57](#)
- OP_CMOVNP
 - [bytecode_disasm.h, 57](#)
- OP_CMOVNS
 - [bytecode_disasm.h, 57](#)
- OP_CMOVNZ
 - [bytecode_disasm.h, 57](#)
- OP_CMOVO
 - [bytecode_disasm.h, 57](#)
- OP_CMOVPP
 - [bytecode_disasm.h, 57](#)
- OP_CMOVSS
 - [bytecode_disasm.h, 57](#)
- OP_CMOVZ
 - [bytecode_disasm.h, 57](#)
- OP_CMP
 - [bytecode_disasm.h, 57](#)
- OP_CMPSB
 - [bytecode_disasm.h, 58](#)
- OP_CMPSD
 - [bytecode_disasm.h, 57](#)
- OP_CMPSW
 - [bytecode_disasm.h, 58](#)
- OP_CMPXCHG
 - [bytecode_disasm.h, 58](#)
- OP_CMPXCHG8B
 - [bytecode_disasm.h, 58](#)
- OP_CPUID
 - [bytecode_disasm.h, 58](#)
- OP_CWDE
 - [bytecode_disasm.h, 57](#)
- OP_DAA
 - [bytecode_disasm.h, 58](#)
- OP_DAS
 - [bytecode_disasm.h, 58](#)
- OP_DEC
 - [bytecode_disasm.h, 58](#)
- OP_DIV
 - [bytecode_disasm.h, 58](#)
- OP_ENTER
 - [bytecode_disasm.h, 58](#)
- OP_F2XM1
 - [bytecode_disasm.h, 61](#)
- OP_FABS
 - [bytecode_disasm.h, 61](#)
- OP_FADD
 - [bytecode_disasm.h, 61](#)
- OP_FADDP
 - [bytecode_disasm.h, 62](#)
- OP_FBLD
 - [bytecode_disasm.h, 62](#)
- OP_FBSTP
 - [bytecode_disasm.h, 62](#)
- OP_FCHS
 - [bytecode_disasm.h, 62](#)
- OP_FCLEX
 - [bytecode_disasm.h, 62](#)
- OP_FCMOVB
 - [bytecode_disasm.h, 62](#)
- OP_FCMOVBE
 - [bytecode_disasm.h, 62](#)
- OP_FCMOVE
 - [bytecode_disasm.h, 62](#)
- OP_FCMOVNB
 - [bytecode_disasm.h, 62](#)
- OP_FCMOVNBE
 - [bytecode_disasm.h, 62](#)
- OP_FCMOVNE
 - [bytecode_disasm.h, 62](#)
- OP_FCMOVNU
 - [bytecode_disasm.h, 62](#)
- OP_FCMOVU
 - [bytecode_disasm.h, 62](#)
- OP_FCOM
 - [bytecode_disasm.h, 62](#)
- OP_FCOMI
 - [bytecode_disasm.h, 62](#)
- OP_FCOMIP
 - [bytecode_disasm.h, 62](#)
- OP_FCOMP
 - [bytecode_disasm.h, 62](#)
- OP_FCOMPP
 - [bytecode_disasm.h, 62](#)
- OP_FCOS
 - [bytecode_disasm.h, 62](#)
- OP_FDECSTP
 - [bytecode_disasm.h, 62](#)
- OP_FDIV
 - [bytecode_disasm.h, 62](#)
- OP_FDIVP
 - [bytecode_disasm.h, 62](#)
- OP_FDIVR
 - [bytecode_disasm.h, 62](#)
- OP_FDIVRP
 - [bytecode_disasm.h, 62](#)
- OP_FFREE
 - [bytecode_disasm.h, 62](#)

OP_FIADD
 bytecode_disasm.h, 62

OP_FICOM
 bytecode_disasm.h, 62

OP_FICOMP
 bytecode_disasm.h, 62

OP_FIDIV
 bytecode_disasm.h, 62

OP_FIDIVR
 bytecode_disasm.h, 62

OP_FILD
 bytecode_disasm.h, 62

OP_FIMUL
 bytecode_disasm.h, 62

OP_FINCSTP
 bytecode_disasm.h, 62

OP_FINIT
 bytecode_disasm.h, 62

OP_FIST
 bytecode_disasm.h, 62

OP_FISTP
 bytecode_disasm.h, 62

OP_FISTTP
 bytecode_disasm.h, 62

OP_FISUB
 bytecode_disasm.h, 62

OP_FISUBR
 bytecode_disasm.h, 63

OP_FLD
 bytecode_disasm.h, 63

OP_FLD1
 bytecode_disasm.h, 63

OP_FLDCW
 bytecode_disasm.h, 63

OP_FLDENV
 bytecode_disasm.h, 63

OP_FLDL2E
 bytecode_disasm.h, 63

OP_FLDL2T
 bytecode_disasm.h, 63

OP_FLDLG2
 bytecode_disasm.h, 63

OP_FLDLN2
 bytecode_disasm.h, 63

OP_FLDPI
 bytecode_disasm.h, 63

OP_FLDZ
 bytecode_disasm.h, 63

OP_FMUL
 bytecode_disasm.h, 63

OP_FMULP
 bytecode_disasm.h, 63

OP_FNOP
 bytecode_disasm.h, 63

OP_FPATAN
 bytecode_disasm.h, 63

OP_FPREM
 bytecode_disasm.h, 63

OP_FPREM1
 bytecode_disasm.h, 63

OP_FPTAN
 bytecode_disasm.h, 63

OP_FPU
 bytecode_disasm.h, 61

OP_FRNDINT
 bytecode_disasm.h, 63

OP_FRSTOR
 bytecode_disasm.h, 63

OP_FSAVE
 bytecode_disasm.h, 63

OP_FSCALE
 bytecode_disasm.h, 63

OP_FSINCOS
 bytecode_disasm.h, 63

OP_FSQRT
 bytecode_disasm.h, 63

OP_FST
 bytecode_disasm.h, 63

OP_FSTCW
 bytecode_disasm.h, 63

OP_FSTENV
 bytecode_disasm.h, 63

OP_FSTP
 bytecode_disasm.h, 63

OP_FSTSW
 bytecode_disasm.h, 63

OP_FSUB
 bytecode_disasm.h, 63

OP_FSUBP
 bytecode_disasm.h, 63

OP_FSUBR
 bytecode_disasm.h, 63

OP_FSUBRP
 bytecode_disasm.h, 63

OP_FTST
 bytecode_disasm.h, 63

OP_FUCOM
 bytecode_disasm.h, 63

OP_FUCOMI
 bytecode_disasm.h, 63

- OP_FUCOMIP
 - bytecode_disasm.h, [63](#)
- OP_FUCOMP
 - bytecode_disasm.h, [63](#)
- OP_FUCOMPP
 - bytecode_disasm.h, [64](#)
- OP_FWAIT
 - bytecode_disasm.h, [58](#)
- OP_FXAM
 - bytecode_disasm.h, [64](#)
- OP_FXCH
 - bytecode_disasm.h, [64](#)
- OP_FXTRACT
 - bytecode_disasm.h, [64](#)
- OP_FYL2X
 - bytecode_disasm.h, [64](#)
- OP_FYL2XP1
 - bytecode_disasm.h, [64](#)
- OP_HLT
 - bytecode_disasm.h, [58](#)
- OP_IDIV
 - bytecode_disasm.h, [58](#)
- OP_IMUL
 - bytecode_disasm.h, [58](#)
- OP_IN
 - bytecode_disasm.h, [58](#)
- OP_INC
 - bytecode_disasm.h, [58](#)
- OP_INSB
 - bytecode_disasm.h, [58](#)
- OP_INSD
 - bytecode_disasm.h, [58](#)
- OP_INSW
 - bytecode_disasm.h, [58](#)
- OP_INT
 - bytecode_disasm.h, [58](#)
- OP_INT3
 - bytecode_disasm.h, [58](#)
- OP_INT0
 - bytecode_disasm.h, [58](#)
- OP_INVDP
 - bytecode_disasm.h, [58](#)
- OP_INVLPD
 - bytecode_disasm.h, [58](#)
- OP_IRET
 - bytecode_disasm.h, [58](#)
- OP_JA
 - bytecode_disasm.h, [58](#)
- OP_JBE
 - bytecode_disasm.h, [58](#)
- OP_JC
 - bytecode_disasm.h, [58](#)
- OP_JECXZ
 - bytecode_disasm.h, [59](#)
- OP_JG
 - bytecode_disasm.h, [59](#)
- OP_JGE
 - bytecode_disasm.h, [59](#)
- OP_JL
 - bytecode_disasm.h, [58](#)
- OP_JLE
 - bytecode_disasm.h, [59](#)
- OP_JMP
 - bytecode_disasm.h, [59](#)
- OP_JNC
 - bytecode_disasm.h, [58](#)
- OP_JNO
 - bytecode_disasm.h, [58](#)
- OP_JNP
 - bytecode_disasm.h, [58](#)
- OP_JNS
 - bytecode_disasm.h, [58](#)
- OP_JNZ
 - bytecode_disasm.h, [58](#)
- OP_JO
 - bytecode_disasm.h, [58](#)
- OP_JP
 - bytecode_disasm.h, [58](#)
- OP_JS
 - bytecode_disasm.h, [58](#)
- OP_JZ
 - bytecode_disasm.h, [58](#)
- OP_LAHF
 - bytecode_disasm.h, [59](#)
- OP_LAR
 - bytecode_disasm.h, [59](#)
- OP_LDS
 - bytecode_disasm.h, [59](#)
- OP_LEA
 - bytecode_disasm.h, [59](#)
- OP_LEAVE
 - bytecode_disasm.h, [59](#)
- OP_LES
 - bytecode_disasm.h, [59](#)
- OP_LFS
 - bytecode_disasm.h, [59](#)
- OP_LGDT
 - bytecode_disasm.h, [59](#)
- OP_LGS
 - bytecode_disasm.h, [59](#)

- OP_LIDT
 - bytecode_disasm.h, [59](#)
- OP_LLDT
 - bytecode_disasm.h, [59](#)
- OP_LODSB
 - bytecode_disasm.h, [59](#)
- OP_LODSD
 - bytecode_disasm.h, [59](#)
- OP_LODSW
 - bytecode_disasm.h, [59](#)
- OP_LOOP
 - bytecode_disasm.h, [59](#)
- OP_LOOPE
 - bytecode_disasm.h, [59](#)
- OP_LOOPNE
 - bytecode_disasm.h, [59](#)
- OP_LSL
 - bytecode_disasm.h, [59](#)
- OP_LSS
 - bytecode_disasm.h, [59](#)
- OP_LTR
 - bytecode_disasm.h, [59](#)
- OP_MOV
 - bytecode_disasm.h, [59](#)
- OP_MOVSBB
 - bytecode_disasm.h, [59](#)
- OP_MOVSD
 - bytecode_disasm.h, [59](#)
- OP_MOVSQ
 - bytecode_disasm.h, [59](#)
- OP_MOVSX
 - bytecode_disasm.h, [59](#)
- OP_MOVZX
 - bytecode_disasm.h, [59](#)
- OP_MUL
 - bytecode_disasm.h, [59](#)
- OP_NEG
 - bytecode_disasm.h, [59](#)
- OP_NOP
 - bytecode_disasm.h, [59](#)
- OP_NOT
 - bytecode_disasm.h, [59](#)
- OP_OR
 - bytecode_disasm.h, [59](#)
- OP_OUT
 - bytecode_disasm.h, [59](#)
- OP_OUTSB
 - bytecode_disasm.h, [60](#)
- OP_OUTSD
 - bytecode_disasm.h, [60](#)
- OP_OUTSW
 - bytecode_disasm.h, [60](#)
- OP_POP
 - bytecode_disasm.h, [60](#)
- OP_POPAD
 - bytecode_disasm.h, [60](#)
- OP_POPFD
 - bytecode_disasm.h, [60](#)
- OP_PREFIX_LOCK
 - bytecode_disasm.h, [59](#)
- OP_PREFIX_REPE
 - bytecode_disasm.h, [60](#)
- OP_PREFIX_REPNB
 - bytecode_disasm.h, [60](#)
- OP_PUSH
 - bytecode_disasm.h, [60](#)
- OP_PUSHAD
 - bytecode_disasm.h, [60](#)
- OP_PUSHFQ
 - bytecode_disasm.h, [60](#)
- OP_RCL
 - bytecode_disasm.h, [60](#)
- OP_RCR
 - bytecode_disasm.h, [60](#)
- OP_RDMSR
 - bytecode_disasm.h, [60](#)
- OP_RDPMC
 - bytecode_disasm.h, [60](#)
- OP_RDTSC
 - bytecode_disasm.h, [60](#)
- OP_RETF
 - bytecode_disasm.h, [60](#)
- OP_RETN
 - bytecode_disasm.h, [60](#)
- OP_ROL
 - bytecode_disasm.h, [60](#)
- OP_ROR
 - bytecode_disasm.h, [60](#)
- OP_RSM
 - bytecode_disasm.h, [60](#)
- OP_SAHF
 - bytecode_disasm.h, [60](#)
- OP_SAR
 - bytecode_disasm.h, [60](#)
- OP_SBB
 - bytecode_disasm.h, [60](#)
- OP_SCASB
 - bytecode_disasm.h, [60](#)
- OP_SCASD
 - bytecode_disasm.h, [60](#)

- OP_SCASW
 - bytecode_disasm.h, 60
- OP_SETA
 - bytecode_disasm.h, 60
- OP_SETBE
 - bytecode_disasm.h, 60
- OP_SETC
 - bytecode_disasm.h, 60
- OP_SETG
 - bytecode_disasm.h, 61
- OP_SETGE
 - bytecode_disasm.h, 61
- OP_SETL
 - bytecode_disasm.h, 61
- OP_SETLE
 - bytecode_disasm.h, 61
- OP_SETNC
 - bytecode_disasm.h, 60
- OP_SETNO
 - bytecode_disasm.h, 60
- OP_SETNP
 - bytecode_disasm.h, 61
- OP_SETNS
 - bytecode_disasm.h, 60
- OP_SETNZ
 - bytecode_disasm.h, 60
- OP_SETO
 - bytecode_disasm.h, 60
- OP_SETP
 - bytecode_disasm.h, 60
- OP_SETS
 - bytecode_disasm.h, 60
- OP_SETZ
 - bytecode_disasm.h, 60
- OP_SGDT
 - bytecode_disasm.h, 61
- OP_SHL
 - bytecode_disasm.h, 61
- OP_SHLD
 - bytecode_disasm.h, 61
- OP_SHR
 - bytecode_disasm.h, 61
- OP_SHRD
 - bytecode_disasm.h, 61
- OP_SIDT
 - bytecode_disasm.h, 61
- OP_SLDT
 - bytecode_disasm.h, 61
- OP_STC
 - bytecode_disasm.h, 61
- OP_STD
 - bytecode_disasm.h, 61
- OP_STI
 - bytecode_disasm.h, 61
- OP_STOSB
 - bytecode_disasm.h, 61
- OP_STOSD
 - bytecode_disasm.h, 61
- OP_STOSW
 - bytecode_disasm.h, 61
- OP_STR
 - bytecode_disasm.h, 61
- OP_SUB
 - bytecode_disasm.h, 61
- OP_SYSCALL
 - bytecode_disasm.h, 61
- OP_SYSENTER
 - bytecode_disasm.h, 61
- OP_SYSEXIT
 - bytecode_disasm.h, 61
- OP_SYSRET
 - bytecode_disasm.h, 61
- OP_TEST
 - bytecode_disasm.h, 61
- OP_UD2
 - bytecode_disasm.h, 61
- OP_VERR
 - bytecode_disasm.h, 61
- OP_VERRW
 - bytecode_disasm.h, 61
- OP_WBINVD
 - bytecode_disasm.h, 61
- OP_WRMSR
 - bytecode_disasm.h, 61
- OP_XADD
 - bytecode_disasm.h, 61
- OP_XCHG
 - bytecode_disasm.h, 61
- OP_XLAT
 - bytecode_disasm.h, 61
- OP_XOR
 - bytecode_disasm.h, 61
- operation_size
 - DIS_fixed, 17
- opt32
 - cli_pe_hook_data, 15
- opt64
 - cli_pe_hook_data, 15
- other
 - DIS_arg, 16

- overlays
 - cli_pe_hook_data, 15
- overlays_sz
 - cli_pe_hook_data, 16
- pdf_flag
 - bytecode_api.h, 27
- pdf_get_dumpedobjid
 - bytecode_api.h, 47
- pdf_get_flags
 - bytecode_api.h, 48
- pdf_get_obj_num
 - bytecode_api.h, 48
- pdf_get_phase
 - bytecode_api.h, 48
- pdf_getobj
 - bytecode_api.h, 48
- pdf_getobjsize
 - bytecode_api.h, 49
- PDF_HOOK_DECLARE
 - bytecode_local.h, 68
- pdf_lookupobj
 - bytecode_api.h, 49
- pdf_objflags
 - bytecode_api.h, 27
- pdf_phase
 - bytecode_api.h, 27
- pdf_set_flags
 - bytecode_api.h, 49
- PE_INVALID_RVA
 - bytecode_api.h, 26
- PE_HOOK_DECLARE
 - bytecode_local.h, 68
- pe_image_data_dir, 18
- pe_image_file_hdr, 18
 - Machine, 19
 - Magic, 19
 - NumberOfSections, 19
 - NumberOfSymbols, 19
 - PointerToSymbolTable, 19
 - SizeOfOptionalHeader, 19
 - TimeStamp, 19
- pe_image_optional_hdr32, 19
 - Checksum, 20
 - FileAlignment, 20
 - ImageBase, 20
 - MajorImageVersion, 20
 - MajorLinkerVersion, 20
 - MajorOperatingSystemVersion, 20
 - MinorImageVersion, 20
 - MinorLinkerVersion, 21
 - MinorOperatingSystemVersion, 21
 - NumberOfRvaAndSizes, 21
 - SectionAlignment, 21
 - SizeOfCode, 21
 - SizeOfInitializedData, 21
 - SizeOfUninitializedData, 21
- pe_image_optional_hdr64, 21
 - Checksum, 22
 - FileAlignment, 22
 - ImageBase, 22
 - MajorImageVersion, 22
 - MajorLinkerVersion, 22
 - MajorOperatingSystemVersion, 22
 - MinorImageVersion, 22
 - MinorLinkerVersion, 22
 - MinorOperatingSystemVersion, 22
 - NumberOfRvaAndSizes, 22
 - SectionAlignment, 22
 - SizeOfCode, 23
 - SizeOfInitializedData, 23
 - SizeOfUninitializedData, 23
- pe_image_section_hdr, 23
 - Name, 23
 - NumberOfLinenumbers, 23
 - NumberOfRelocations, 23
 - PointerToLinenumbers, 23
 - PointerToRawData, 24
 - PointerToRelocations, 24
 - SizeOfRawData, 24
- pe_rawaddr
 - bytecode_api.h, 50
- PE_UNPACKER_DECLARE
 - bytecode_local.h, 69
- PointerToLinenumbers
 - pe_image_section_hdr, 23
- PointerToRawData
 - pe_image_section_hdr, 24
- PointerToRelocations
 - pe_image_section_hdr, 24
- PointerToSymbolTable
 - pe_image_file_hdr, 19
- raw
 - cli_exe_section, 14
- read
 - bytecode_api.h, 50
- read_number
 - bytecode_api.h, 50
- readPESectionName

- bytecode_local.h, 88
- readRVA
 - bytecode_local.h, 88
- reg
 - DIS_arg, 16
- res_addr
 - cli_exe_info, 13
- rsz
 - cli_exe_section, 14
- rva
 - cli_exe_section, 14
- scale
 - DIS_mem_arg, 18
- scale_reg
 - DIS_mem_arg, 18
- section
 - cli_exe_info, 13
- SectionAlignment
 - pe_image_optional_hdr32, 21
 - pe_image_optional_hdr64, 22
- seek
 - bytecode_api.h, 51
- SEEK_CUR
 - bytecode_api.h, 27
- SEEK_END
 - bytecode_api.h, 27
- SEEK_SET
 - bytecode_api.h, 27
- segment
 - DIS_fixed, 17
- setvirusname
 - bytecode_api.h, 51
- SIGNATURES_DECL_BEGIN
 - bytecode_local.h, 69
- SIGNATURES_DECL_END
 - bytecode_local.h, 69
- SIGNATURES_DEF_BEGIN
 - bytecode_local.h, 69
- SIGNATURES_END
 - bytecode_local.h, 69
- SIZEB
 - bytecode_disasm.h, 56
- SIZED
 - bytecode_disasm.h, 56
- SIZEF
 - bytecode_disasm.h, 56
- SizeOfCode
 - pe_image_optional_hdr32, 21
 - pe_image_optional_hdr64, 23
- SizeOfInitializedData
 - pe_image_optional_hdr32, 21
 - pe_image_optional_hdr64, 23
- SizeOfOptionalHeader
 - pe_image_file_hdr, 19
- SizeOfRawData
 - pe_image_section_hdr, 24
- SizeOfUninitializedData
 - pe_image_optional_hdr32, 21
 - pe_image_optional_hdr64, 23
- SIZEPTR
 - bytecode_disasm.h, 56
- SIZEQ
 - bytecode_disasm.h, 56
- SIZET
 - bytecode_disasm.h, 56
- SIZEW
 - bytecode_disasm.h, 56
- TARGET
 - bytecode_local.h, 70
- test1
 - bytecode_api.h, 51
- test2
 - bytecode_api.h, 52
- TimeStampStamp
 - pe_image_file_hdr, 19
- uraw
 - cli_exe_section, 14
- ursz
 - cli_exe_section, 14
- urva
 - cli_exe_section, 14
- uvsz
 - cli_exe_section, 14
- version_compare
 - bytecode_api.h, 52
- VIRUSNAME_PREFIX
 - bytecode_local.h, 70
- VIRUSNAMES
 - bytecode_local.h, 70
- vsz
 - cli_exe_section, 14
- write
 - bytecode_api.h, 52
- x86_opcode
 - DIS_fixed, 17

X86OPS

 bytecode_disasm.h, [56](#)

X86REGS

 bytecode_disasm.h, [64](#)