

# Reference Manual

Generated by Doxygen 1.8.6

Tue Feb 25 2014 15:05:00

## Contents

<b>1</b>	<b>Module Documentation</b>	<b>1</b>
1.1	Abstract Data Types	1
1.1.1	Detailed Description	2
1.1.2	Function Documentation	2
1.2	Bytecode Configuration	9
1.2.1	Detailed Description	9
1.2.2	Macro Definition Documentation	9
1.2.3	Enumeration Type Documentation	11
1.3	Debugging	13
1.3.1	Detailed Description	13
1.3.2	Function Documentation	13
1.4	Disassembly	16
1.4.1	Detailed Description	16
1.4.2	Function Documentation	16
1.5	Engine Queries	17
1.5.1	Detailed Description	17
1.5.2	Function Documentation	17
1.6	Environment	19
1.6.1	Detailed Description	19
1.6.2	Function Documentation	19
1.7	File Operations	23
1.7.1	Detailed Description	23
1.7.2	Enumeration Type Documentation	23
1.7.3	Function Documentation	23
1.8	Global Variables	27
1.8.1	Detailed Description	27
1.8.2	Variable Documentation	27
1.9	JavaScript Normalization	28
1.9.1	Detailed Description	28
1.9.2	Function Documentation	28
1.10	Icon Matcher	29
1.10.1	Detailed Description	29
1.10.2	Function Documentation	29
1.11	Math Operation	30
1.11.1	Detailed Description	30
1.11.2	Function Documentation	30
1.12	PDF Handling	32
1.12.1	Detailed Description	32

1.12.2	Enumeration Type Documentation	32
1.12.3	Function Documentation	32
1.13	PE Operations	37
1.13.1	Detailed Description	38
1.13.2	Function Documentation	38
1.14	Scan Control	46
1.14.1	Detailed Description	46
1.14.2	Function Documentation	46
1.15	String Operations	48
1.15.1	Detailed Description	48
1.15.2	Function Documentation	48
<b>2</b>	<b>Data Structure Documentation</b>	<b>51</b>
2.1	cli_exe_info Struct Reference	51
2.1.1	Detailed Description	51
2.1.2	Field Documentation	51
2.2	cli_exe_section Struct Reference	51
2.2.1	Detailed Description	52
2.2.2	Field Documentation	52
2.3	cli_pe_hook_data Struct Reference	52
2.3.1	Detailed Description	53
2.3.2	Field Documentation	53
2.4	DIS_arg Struct Reference	53
2.4.1	Detailed Description	53
2.4.2	Field Documentation	54
2.5	DIS_fixed Struct Reference	54
2.5.1	Detailed Description	54
2.5.2	Field Documentation	54
2.6	DIS_mem_arg Struct Reference	55
2.6.1	Detailed Description	55
2.6.2	Field Documentation	55
2.7	DISASM_RESULT Struct Reference	55
2.7.1	Detailed Description	55
2.8	pe_image_data_dir Struct Reference	55
2.8.1	Detailed Description	55
2.9	pe_image_file_hdr Struct Reference	55
2.9.1	Detailed Description	56
2.9.2	Field Documentation	56
2.10	pe_image_optional_hdr32 Struct Reference	56
2.10.1	Detailed Description	57

2.10.2	Field Documentation	57
2.11	pe_image_optional_hdr64 Struct Reference	58
2.11.1	Detailed Description	58
2.11.2	Field Documentation	58
2.12	pe_image_section_hdr Struct Reference	59
2.12.1	Detailed Description	59
2.12.2	Field Documentation	59
<b>3</b>	<b>File Documentation</b>	<b>60</b>
3.1	bytecode_api.h File Reference	60
3.1.1	Enumeration Type Documentation	62
3.1.2	Function Documentation	62
3.2	bytecode_disasm.h File Reference	63
3.2.1	Enumeration Type Documentation	65
3.3	bytecode_execs.h File Reference	72
3.4	bytecode_local.h File Reference	72
3.4.1	Macro Definition Documentation	74
3.4.2	Function Documentation	74
3.5	bytecode_pe.h File Reference	74
	<b>Index</b>	<b>75</b>

## 1 Module Documentation

### 1.1 Abstract Data Types

#### Functions

- void \* [malloc](#) (uint32\_t size)
- 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)
- const 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 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`)

### 1.1.1 Detailed Description

### 1.1.2 Function Documentation

#### 1.1.2.1 `int32_t buffer_pipe_done ( int32_t id )`

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

##### Parameters

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

##### Returns

0 on success

#### 1.1.2.2 `int32_t buffer_pipe_new ( uint32_t size )`

Creates a new pipe with the specified buffer size

##### Parameters

<code>in</code>	<code>size</code>	size of buffer
-----------------	-------------------	----------------

##### Returns

ID of newly created `buffer_pipe`

#### 1.1.2.3 `int32_t buffer_pipe_new_fromfile ( uint32_t pos )`

Creates a new pipe with the specified buffer size w/ tied input to the current file, at the specified position.

##### Parameters

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

##### Returns

ID of newly created `buffer_pipe`

#### 1.1.2.4 `uint32_t buffer_pipe_read_avail ( int32_t id )`

Returns the amount of bytes available to read.

**Parameters**

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

**Returns**

amount of bytes available to read

**1.1.2.5 `const uint8_t* buffer_pipe_read_get ( int32_t id, uint32_t amount )`**

Returns a pointer to the buffer for reading. The 'amount' parameter should be obtained by a call to [buffer\\_pipe\\_read\\_avail\(\)](#).

**Parameters**

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

**Returns**

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

**1.1.2.6 `int32_t buffer_pipe_read_stopped ( int32_t id, uint32_t amount )`**

Updates read cursor in `buffer_pipe`.

**Parameters**

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

**Returns**

0 on success

**1.1.2.7 `uint32_t buffer_pipe_write_avail ( int32_t id )`**

Returns the amount of bytes available for writing.

**Parameters**

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

**Returns**

amount of bytes available for writing

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

Returns pointer to writable buffer. The 'size' parameter should be obtained by a call to [buffer\\_pipe\\_write\\_avail\(\)](#).

**Parameters**

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

**1.1.2.9 `int32_t buffer_pipe_write_stopped ( int32_t id, uint32_t amount )`**

Updates the write cursor in `buffer_pipe`.

**Parameters**

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

**Returns**

0 on success

**1.1.2.10 `int32_t` *hashset\_add* ( `int32_t` *hs*, `uint32_t` *key* )**

Add a new 32-bit key to the hashset.

**Parameters**

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

**Returns**

0 on success

**1.1.2.11 `int32_t` *hashset\_contains* ( `int32_t` *hs*, `uint32_t` *key* )**

Returns whether the hashset contains the specified key.

**Parameters**

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

**Returns**

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

**1.1.2.12 `int32_t` *hashset\_done* ( `int32_t` *id* )**

Deallocates the memory used by the specified hashset. 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>in</i>	<i>id</i>	ID of hashset (from <i>hashset_new</i> )
-----------	-----------	--

**Returns**

0 on success

**1.1.2.13 `int32_t` *hashset\_empty* ( `int32_t` *id* )**

Returns whether the hashset is empty.

**Parameters**

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

**Returns**

0 on success

**1.1.2.14 `int32_t hashset_new ( void )`**

Creates a new hashset and returns its id.

**Returns**

ID for new hashset

**1.1.2.15 `int32_t hashset_remove ( int32_t hs, uint32_t key )`**

Remove a 32-bit key from the hashset.

**Parameters**

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

**Returns**

0 on success

**1.1.2.16 `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.

**Parameters**

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

**Returns**

0 on success.

**1.1.2.17 `int32_t inflate_init ( int32_t from_buffer, int32_t to_buffer, int32_t windowBits )`**

Initializes inflate data structures for decompressing data 'from\_buffer' and writing uncompressed data 'to\_buffer'.

**Parameters**

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

**Returns**

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

**1.1.2.18 `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.



**Parameters**

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

**Returns**

0 on success, zlib error code otherwise

**1.1.2.19 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.

**Parameters**

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

**Returns**

pointer to allocated memory

**1.1.2.20 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>in</i>	<i>id</i>	id of table
<i>in</i>	<i>key</i>	key
<i>in</i>	<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

**1.1.2.21 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>in</i>	<i>id</i>	id of map
-----------	-----------	-----------

**Returns**

0 - success  
 -1 - invalid map

**1.1.2.22 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**

in	<i>id</i>	id of map
in	<i>key</i>	key
in	<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

**1.1.2.23 uint8\_t\* map\_getvalue ( int32\_t id, int32\_t size )**

Returns the value obtained during last map\_find.

**Parameters**

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

**Returns**

value

**1.1.2.24 int32\_t map\_getvaluesize ( int32\_t id )**

Returns the size of value obtained during last map\_find.

**Parameters**

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

**Returns**

size of value

**1.1.2.25 int32\_t map\_new ( int32\_t keysize, int32\_t valuesize )**

Creates a new map and returns its id.

**Parameters**

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

**Returns**

ID of new map

**1.1.2.26 int32\_t map\_remove ( const uint8\_t \* key, int32\_t ksize, int32\_t id )**

Remove an element from the map.

**Parameters**

in	<i>id</i>	id of map
in	<i>key</i>	key
in	<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

1.1.2.27 `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**

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

**Returns**

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

## 1.2 Bytecode Configuration

### Macros

- `#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 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 SIGNATURES_DEF_END };`

### Enumerations

- `enum BytecodeKind {`  
`BC_GENERIC = 0, BC_STARTUP = 1, BC_LOGICAL = 256, BC_PE_UNPACKER,`  
`BC_PDF, BC_PE_ALL }`
- `enum FunctionalityLevels {`  
`FUNC_LEVEL_096 = 51, FUNC_LEVEL_096_1 = 53, FUNC_LEVEL_096_2 = 54, FUNC_LEVEL_096_3`  
`= 55,`  
`FUNC_LEVEL_096_4 = 56, FUNC_LEVEL_096_5 = 58, FUNC_LEVEL_097 = 60, FUNC_LEVEL_097_1 =`  
`61,`  
`FUNC_LEVEL_097_2 = 62, FUNC_LEVEL_097_3 = 63, FUNC_LEVEL_097_4 = 64, FUNC_LEVEL_097_5`  
`= 65,`  
`FUNC_LEVEL_097_6 = 67, FUNC_LEVEL_097_7 = 68, FUNC_LEVEL_097_8 = 69, FUNC_LEVEL_098 =`  
`74,`  
`FUNC_LEVEL_098_1 = 76, FUNC_LEVEL_098_2 = 78 }`

#### 1.2.1 Detailed Description

#### 1.2.2 Macro Definition Documentation

##### 1.2.2.1 `#define COPYRIGHT( c ) const char *const __Copyright = (c);`

Defines an alternative copyright for this bytecode.

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

##### 1.2.2.2 `#define DECLARE_SIGNATURE( name )`

**Value:**

```
const char *name##_sig;\n__Signature name;
```

Declares a name for a subsignature.

**1.2.2.3** `#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 [FunctionalityLevels](#) enumeration here.

**1.2.2.4** `#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 [FunctionalityLevels](#) enumeration here.

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

Define IconGroup1 for logical signature.

See logical signature documentation for what it is.

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

Define IconGroup2 for logical signature.

See logical signature documentation for what it is.

**1.2.2.7** `#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 the logical signature is evaluated AFTER these hooks run.

This hook is called several times, use [pdf\\_get\\_phase\(\)](#) to find out in which phase you got called.

**1.2.2.8** `#define PE_HOOK_DECLARE const uint16_t __clambc_kind = BC_PE_ALL;`

Make the current bytecode a PE hook.

Bytecode will be called once the logical signature trigger matches (or always if there is none), and if 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).

**1.2.2.9** `#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).

**1.2.2.10** `#define SIGNATURES_DECL_BEGIN struct __Signatures {`

Marks the beginning of the subsignature name declaration section.

**1.2.2.11** `#define SIGNATURES_DECL_END };`

Marks the end of the subsignature name declaration section.

**1.2.2.12** `#define SIGNATURES_DEF_BEGIN`

**Value:**

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

Marks the beginning of subsignature pattern definitions.

See Also

[SIGNATURES\\_DECL\\_BEGIN](#)

**1.2.2.13 #define SIGNATURES\_DEF\_END ;**

Marks the end of the subsignature pattern definitions.

Alternative: SIGNATURES\_END

**1.2.2.14 #define TARGET( tgt ) const unsigned short \_\_Target = (tgt);**

Defines the ClamAV file target.

Parameters

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

**1.2.2.15 #define VIRUSNAME\_PREFIX( name ) const char \_\_clambc\_virusname\_prefix[] = name;**

Declares the virusname prefix.

Parameters

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

**1.2.2.16 #define VIRUSNAMES( ... ) const char \*const \_\_clambc\_virusnames[] = { \_\_VA\_ARGS\_\_ };**

Declares all the virusnames that this bytecode can report.

Parameters

in	...	a comma-separated list of strings interpreted as virusnames
----	-----	---

**1.2.3 Enumeration Type Documentation****1.2.3.1 enum BytecodeKind**

Specifies the bytecode type and how ClamAV executes it

Enumerator

**BC\_GENERIC** generic bytecode, not tied a specific hook

**BC\_STARTUP** triggered at startup, only one is allowed per ClamAV startup

**BC\_LOGICAL** executed on a logical trigger

**BC\_PE\_UNPACKER** specifies a PE unpacker, executed on PE files on a logical trigger

**BC\_PDF** specifies a PDF hook, executes at a predetermined point of PDF parsing for PDF files

**BC\_PE\_ALL** specifies a PE hook, executes at a predetermined point in PE parsing for PE files, both packed and unpacked files

**1.2.3.2 enum FunctionalityLevels**

LibClamAV functionality level constants

Enumerator

**FUNC\_LEVEL\_096** LibClamAV release 0.96.0: bytecode engine released

**FUNC\_LEVEL\_096\_1** LibClamAV release 0.96.1: logical signature use of VI/macros requires this minimum functionality level

**FUNC\_LEVEL\_096\_2** LibClamAV release 0.96.2: PDF Hooks require this minimum level

**FUNC\_LEVEL\_096\_3** LibClamAV release 0.96.3: BC\_PE\_ALL bytecodes require this minimum level

**FUNC\_LEVEL\_096\_4** LibClamAV release 0.96.4: minimum recommended engine version, older versions have quadratic load time

***FUNC\_LEVEL\_096\_5*** LibClamAV release 0.96.5  
***FUNC\_LEVEL\_097*** LibClamAV release 0.97.0: older bytecodes may incorrectly use 57  
***FUNC\_LEVEL\_097\_1*** LibClamAV release 0.97.1  
***FUNC\_LEVEL\_097\_2*** LibClamAV release 0.97.2  
***FUNC\_LEVEL\_097\_3*** LibClamAV release 0.97.3  
***FUNC\_LEVEL\_097\_4*** LibClamAV release 0.97.4  
***FUNC\_LEVEL\_097\_5*** LibClamAV release 0.97.5  
***FUNC\_LEVEL\_097\_6*** LibClamAV release 0.97.6  
***FUNC\_LEVEL\_097\_7*** LibClamAV release 0.97.7  
***FUNC\_LEVEL\_097\_8*** LibClamAV release 0.97.8  
***FUNC\_LEVEL\_098*** LibClamAV release 0.98.0  
***FUNC\_LEVEL\_098\_1*** LibClamAV release 0.98.1  
***FUNC\_LEVEL\_098\_2*** LibClamAV release 0.98.2

## 1.3 Debugging

### Functions

- uint32\_t [debug\\_print\\_str](#) (const uint8\_t \*str, uint32\_t len)
- uint32\_t [debug\\_print\\_uint](#) (uint32\_t a)
- 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)
- void [debug](#) (...) \_\_attribute\_\_((overloadable))
- 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)

#### 1.3.1 Detailed Description

#### 1.3.2 Function Documentation

##### 1.3.2.1 `debug ( const char * str ) [static]`

Prints `str` to clamscan's `-debug` output. This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

##### Parameters

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

##### 1.3.2.2 `debug ( const uint8_t * str ) [static]`

Prints `str` to clamscan's `-debug` output. This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

##### Parameters

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

##### 1.3.2.3 `debug ( uint32_t a ) [static]`

Prints `a` integer to clamscan's `-debug` output. This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

##### Parameters

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

##### 1.3.2.4 `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.

##### See Also

```
debug(const char * str),
debug(const uint8_t* str),
debug(uint32_t a)
```



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

Prints a debug message string.

## Parameters

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

## Returns

0

1.3.2.6 `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

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

## Returns

0

1.3.2.7 `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

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

## Returns

0

1.3.2.8 `uint32_t debug_print_uint ( uint32_t a )`

Prints a number as a debug message. This is similar to `debug_print_str_nonl`.

## Parameters

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

## Returns

0

## 1.4 Disassembly

### Data Structures

- struct [DIS\\_mem\\_arg](#)
- struct [DIS\\_arg](#)
- struct [DIS\\_fixed](#)

### Functions

- uint32\_t [disasm\\_x86](#) (struct [DISASM\\_RESULT](#) \*result, uint32\_t len)
- static force\_inline uint32\_t [DisassembleAt](#) (struct [DIS\\_fixed](#) \*result, uint32\_t offset, uint32\_t len)

#### 1.4.1 Detailed Description

#### 1.4.2 Function Documentation

##### 1.4.2.1 uint32\_t disasm\_x86 ( struct [DISASM\\_RESULT](#) \* result, uint32\_t len )

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

#### Parameters

out	<i>result</i>	pointer to struct holding result
in	<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](#)

##### 1.4.2.2 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.

#### 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

## 1.5 Engine Queries

### Functions

- 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 [running\\_on\\_jit](#) (void)
- 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)

### 1.5.1 Detailed Description

### 1.5.2 Function Documentation

#### 1.5.2.1 static force\_inline uint32\_t count\_match ( \_\_Signature *sig* ) [static]

Returns how many times the specified signature matched.

#### Parameters

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

#### Returns

number of times this subsignature matched in the entire file

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

#### 1.5.2.2 uint32\_t engine\_db\_options ( void )

Returns the current engine's db options.

#### Returns

CL\_DB\_\* flags

#### 1.5.2.3 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.

#### 1.5.2.4 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.

### 1.5.2.5 uint32\_t engine\_scan\_options ( void )

Returns the current engine's scan options.

Returns

CL\_SCAN\* flags

### 1.5.2.6 static force\_inline uint32\_t match\_location ( \_\_Signature sig, uint32\_t goback ) [static]

Returns the offset of the match.

Parameters

in	<i>sig</i>	- Signature
in	<i>goback</i>	- max length of signature

Returns

offset of match

### 1.5.2.7 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.

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

Parameters

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

Returns

>=0 - offset of match

-1 - no match

### 1.5.2.8 static force\_inline uint32\_t matches ( \_\_Signature sig ) [static]

Returns whether the specified subsignature has matched at least once.

Parameters

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

Returns

1 if subsignature one or more times, 0 otherwise

### 1.5.2.9 int32\_t running\_on\_jit ( void )

Returns whether running on JIT. As side-effect it disables interp / JIT comparisons in test mode (errors are still checked)

Returns

1 - running on JIT

0 - running on ClamAV interpreter

## 1.6 Environment

### Functions

- 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)
- bool [\\_\\_is\\_bigendian](#) (void) [\\_\\_attribute\\_\\_\(\(const\)\)](#) [\\_\\_attribute\\_\\_\(\(nothrow\)\)](#)
- static uint32\_t force\_inline [le32\\_to\\_host](#) (uint32\_t v)
- static uint32\_t force\_inline [be32\\_to\\_host](#) (uint32\_t v)
- static uint64\_t force\_inline [le64\\_to\\_host](#) (uint64\_t v)
- static uint64\_t force\_inline [be64\\_to\\_host](#) (uint64\_t v)
- static uint16\_t force\_inline [le16\\_to\\_host](#) (uint16\_t v)
- static uint16\_t force\_inline [be16\\_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)

#### 1.6.1 Detailed Description

#### 1.6.2 Function Documentation

##### 1.6.2.1 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

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).

##### 1.6.2.2 static uint16\_t force\_inline [be16\\_to\\_host](#) ( uint16\_t v ) [static]

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

#### Parameters

in	v	16-bit integer as read from a file
----	---	------------------------------------

#### Returns

integer converted to host's endianness

##### 1.6.2.3 static uint32\_t force\_inline [be32\\_to\\_host](#) ( uint32\_t v ) [static]

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

**Parameters**

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

**Returns**

integer converted to host's endianness

**1.6.2.4** `static uint64_t force_inline be64_to_host ( uint64_t v ) [static]`

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

**Parameters**

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

**Returns**

integer converted to host's endianness

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

**Returns**

0 - no match  
1 - match

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

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

**Parameters**

<i>in</i>	<i>buff</i>	pointer to buffer
-----------	-------------	-------------------

**Returns**

16-bit little-endian integer converted to host endianness

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

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

**Parameters**

<i>in</i>	<i>buff</i>	pointer to buffer
-----------	-------------	-------------------

**Returns**

32-bit little-endian integer converted to host endianness

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

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

## Parameters

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

## 1.6.2.9 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

in	reason	- why the bytecode had to be disabled
in	len	- length of reason
in	cond	- condition

## Returns

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

## 1.6.2.10 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

in	reason	- why the JIT had to be disabled
in	len	- length of reason
in	cond	- condition

## Returns

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

## 1.6.2.11 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	env	- the full environment
in	len	- size of env

## Returns

- 0

## 1.6.2.12 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.



**Parameters**

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

**Returns**

integer converted to host's endianness

**1.6.2.13** `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.

**Parameters**

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

**Returns**

integer converted to host's endianness

**1.6.2.14** `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.

**Parameters**

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

**Returns**

integer converted to host's endianness

**1.6.2.15** `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**

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

**Returns**

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

## 1.7 File Operations

### Enumerations

- enum { [SEEK\\_SET](#) =0, [SEEK\\_CUR](#), [SEEK\\_END](#) }

### Functions

- int32\_t [read](#) (uint8\_t \*data, int32\_t size)
- int32\_t [write](#) (uint8\_t \*data, int32\_t size)
- int32\_t [seek](#) (int32\_t pos, uint32\_t whence)
- int32\_t [file\\_find](#) (const uint8\_t \*data, uint32\_t len)
- int32\_t [file\\_byteat](#) (uint32\_t offset)
- int32\_t [fill\\_buffer](#) (uint8\_t \*buffer, uint32\_t len, uint32\_t filled, uint32\_t cursor, uint32\_t fill)
- int32\_t [read\\_number](#) (uint32\_t radix)
- int32\_t [file\\_find\\_limit](#) (const uint8\_t \*data, uint32\_t len, int32\_t maxpos)
- int32\_t [get\\_file\\_reliability](#) (void)
- static force\_inline uint32\_t [getFilesize](#) (void)

#### 1.7.1 Detailed Description

#### 1.7.2 Enumeration Type Documentation

##### 1.7.2.1 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

#### 1.7.3 Function Documentation

##### 1.7.3.1 int32\_t file\_byteat ( uint32\_t offset )

Read a single byte from current file

### Parameters

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

### Returns

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

##### 1.7.3.2 int32\_t file\_find ( const uint8\_t \* data, uint32\_t len )

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

### Parameters

in	<i>data</i>	the sequence of bytes to look for
in	<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

1.7.3.3 `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
in	<i>len</i>	length of <i>data</i> , cannot be more than 1024
in	<i>maxpos</i>	maximum position to look for a match, note that this is 1 byte after the end of last possible match: $\text{match\_pos} + \text{len} < \text{maxpos}$

## Returns

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

1.7.3.4 `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.

## 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.

1.7.3.5 `int32_t get_file_reliability ( void )`

Get file reliability flag, higher value means less reliable. When >0 import tables and such are not reliable

## Returns

0 - normal

1 - embedded PE

2 - unpacker created file (not impl. yet)

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

Returns the currently scanned file's size.

## Returns

file size as 32-bit unsigned integer

1.7.3.7 `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	<i>size</i>	amount of bytes to read
----	-------------	-------------------------

<i>out</i>	<i>data</i>	pointer to buffer where data is read into
------------	-------------	---

**Returns**

amount read.

**1.7.3.8 int32\_t read\_number ( uint32\_t radix )**

Reads a number in the specified radix starting from the current position. Non-numeric characters are ignored.

**Parameters**

<i>in</i>	<i>radix</i>	10 or 16
-----------	--------------	----------

**Returns**

the number read

**1.7.3.9 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**

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

**Returns**

absolute position in file

**1.7.3.10 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

## 1.8 Global Variables

### Variables

- `const uint32_t __clambc_match_counts [64]`  
*This is a low-level variable, use the Macros in [bytecode\\_local.h](#) instead to access it.*
- `const uint32_t __clambc_match_offsets [64]`  
*This is a low-level variable, use the Macros in [bytecode\\_local.h](#) instead to access it.*
- `const struct cli_pe_hook_data __clambc_pedata`
- `const uint32_t __clambc_filesize [1]`
- `const uint16_t __clambc_kind`

### 1.8.1 Detailed Description

### 1.8.2 Variable Documentation

#### 1.8.2.1 `const uint32_t __clambc_filesize[1]`

File size (max 4G).

#### 1.8.2.2 `const uint16_t __clambc_kind`

Kind of the bytecode, affects LibClamAV usage

#### 1.8.2.3 `const uint32_t __clambc_match_counts[64]`

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

Logical signature match counts

#### 1.8.2.4 `const uint32_t __clambc_match_offsets[64]`

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

Logical signature match offsets

#### 1.8.2.5 `const struct cli_pe_hook_data __clambc_pedata`

PE data, if this is a PE hook.

## 1.9 JavaScript Normalization

### Functions

- `int32_t jsnorm_init (int32_t from_buffer)`
- `int32_t jsnorm_process (int32_t id)`
- `int32_t jsnorm_done (int32_t id)`

#### 1.9.1 Detailed Description

#### 1.9.2 Function Documentation

##### 1.9.2.1 `int32_t jsnorm_done ( int32_t id )`

Flushes JS normalizer.

#### Parameters

<code>in</code>	<code>id</code>	ID of js normalizer to flush
-----------------	-----------------	------------------------------

#### Returns

0 on success, <0 on failure

##### 1.9.2.2 `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.

#### Parameters

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

#### Returns

ID of JS normalizer, <0 on failure

##### 1.9.2.3 `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.

#### Parameters

<code>in</code>	<code>id</code>	ID of JS normalizer
-----------------	-----------------	---------------------

#### Returns

0 on success, <0 on failure

## 1.10 Icon Matcher

### Functions

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

#### 1.10.1 Detailed Description

#### 1.10.2 Function Documentation

##### 1.10.2.1 `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.

#### Parameters

<code>in</code>	<code>group1</code>	- same as GROUP1 in LDB signatures
<code>in</code>	<code>group1_len</code>	- length of <code>group1</code>
<code>in</code>	<code>group2</code>	- same as GROUP2 in LDB signatures
<code>in</code>	<code>group2_len</code>	- length of <code>group2</code>

#### Returns

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



## 1.11 Math Operation

### Functions

- `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`)

#### 1.11.1 Detailed Description

#### 1.11.2 Function Documentation

##### 1.11.2.1 `int32_t icos ( int32_t a, int32_t b, int32_t c )`

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

#### Parameters

<code>in</code>	<code>a</code>	integer
<code>in</code>	<code>b</code>	integer
<code>in</code>	<code>c</code>	integer

#### Returns

$c \cdot \sin(a/b)$

##### 1.11.2.2 `uint32_t iexp ( int32_t a, int32_t b, int32_t c )`

Returns  $\exp(a/b) \cdot c$

#### Parameters

<code>in</code>	<code>a</code>	integer
<code>in</code>	<code>b</code>	integer
<code>in</code>	<code>c</code>	integer

#### Returns

$c \cdot \exp(a/b)$

##### 1.11.2.3 `int32_t ilog2 ( uint32_t a, uint32_t b )`

Returns  $2^{26} \cdot \log_2(a/b)$

#### Parameters

<code>in</code>	<code>a</code>	input
<code>in</code>	<code>b</code>	input

#### Returns

$2^{26} \cdot \log_2(a/b)$

##### 1.11.2.4 `int32_t ipow ( int32_t a, int32_t b, int32_t c )`

Returns  $c \cdot a^b$ .

**Parameters**

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

**Returns**

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

**1.11.2.5 int32\_t isin ( int32\_t *a*, int32\_t *b*, int32\_t *c* )**

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

**Parameters**

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

**Returns**

$c * \sin(a/b)$

## 1.12 PDF Handling

### Enumerations

- enum `pdf_phase` { , `PDF_PHASE_PARSED`, `PDF_PHASE_POSTDUMP`, `PDF_PHASE_END`, `PDF_PHASE_PRE` }
- enum `pdf_flag`
- enum `pdf_objflags`

### Functions

- `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)
- `const uint8_t * pdf_getobj` (int32\_t objidx, uint32\_t amount)
- `int32_t pdf_getobjid` (int32\_t objidx)
- `int32_t pdf_getobjflags` (int32\_t objidx)
- `int32_t pdf_setobjflags` (int32\_t objidx, int32\_t flags)
- `int32_t pdf_get_offset` (int32\_t objidx)
- `int32_t pdf_get_phase` (void)
- `int32_t pdf_get_dumpedobjid` (void)

#### 1.12.1 Detailed Description

#### 1.12.2 Enumeration Type Documentation

##### 1.12.2.1 enum `pdf_flag`

PDF flags

##### 1.12.2.2 enum `pdf_objflags`

PDF obj flags

##### 1.12.2.3 enum `pdf_phase`

Phase of PDF parsing used for PDF Hooks

#### Enumerator

- `PDF_PHASE_PARSED`** after parsing a PDF, object flags can be set etc.
- `PDF_PHASE_POSTDUMP`** after an obj was dumped and scanned
- `PDF_PHASE_END`** after the pdf scan finished
- `PDF_PHASE_PRE`** before pdf is parsed at all

#### 1.12.3 Function Documentation

##### 1.12.3.1 `int32_t pdf_get_dumpedobjid` ( void )

Return the currently dumped obj index. Valid only in `PDF_PHASE_POSTDUMP`.

#### Returns

- `>=0` - object index
- `-1` - invalid phase

1.12.3.2 `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

1.12.3.3 `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

1.12.3.4 `int32_t pdf_get_offset ( int32_t objidx )`

Return the object's offset in the PDF.

## Parameters

<i>in</i>	<i>objidx</i>	- object index (from 0)
-----------	---------------	-------------------------

## Returns

-1 - object index invalid  
 >=0 - offset

1.12.3.5 `int32_t pdf_get_phase ( void )`

Return an 'enum pdf\_phase'. Identifies at which phase this bytecode was called.

## Returns

the current [pdf\\_phase](#)

1.12.3.6 `const uint8_t* pdf_getobj ( int32_t objidx, uint32_t amount )`

Return the undecoded object. Meant only for reading, write modifies the fmap buffer, so avoid!

## Parameters

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

## Returns

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

1.12.3.7 `int32_t pdf_getobjflags ( int32_t objidx )`

Return the object flags for the specified object index.

**Parameters**

<i>in</i>	<i>objidx</i>	- object index (from 0)
-----------	---------------	-------------------------

**Returns**

- 1 - object index invalid
- >=0 - object flags

**1.12.3.8 int32\_t pdf\_getobjid ( int32\_t objidx )**

Return the object id for the specified object index.

**Parameters**

<i>in</i>	<i>objidx</i>	- object index (from 0)
-----------	---------------	-------------------------

**Returns**

- 1 - object index invalid
- >=0 - object id (obj id << 8 | generation id)

**1.12.3.9 uint32\_t pdf\_getobjsize ( int32\_t objidx )**

Return the size of the specified PDF obj.

**Parameters**

<i>in</i>	<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

**1.12.3.10 int32\_t pdf\_lookupobj ( uint32\_t id )**

Lookup pdf object with specified id.

**Parameters**

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

**Returns**

- 1 - if object id doesn't exist
- >=0 - object index

**1.12.3.11 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.

**Parameters**

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

**Returns**

- 0 - success -1 - invalid phase

#### 1.12.3.12 int32\_t pdf\_setobjflags ( int32\_t *objidx*, int32\_t *flags* )

Sets the object flags for the specified object index. This can be used to force dumping of a certain obj, by setting the OBJ\_FORCEDUMP flag for example.

**Parameters**

<i>in</i>	<i>objidx</i>	- object index (from 0)
<i>in</i>	<i>flags</i>	- value to set flags

**Returns**

-1 - object index invalid  
≥0 - flags set

## 1.13 PE Operations

### Data Structures

- struct [cli\\_exe\\_section](#)
- struct [cli\\_exe\\_info](#)
- 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](#)

### Functions

- uint32\_t [pe\\_rawaddr](#) (uint32\_t rva)
- int32\_t [get\\_pe\\_section](#) (struct [cli\\_exe\\_section](#) \*section, uint32\_t num)
- static force\_inline bool [hasExeInfo](#) (void)
- static force\_inline bool [hasPEInfo](#) (void)
- static force\_inline bool [isPE64](#) (void)
- static 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)
- 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 [getPEDirRVA](#) (unsigned n)
- static force\_inline uint32\_t [getPEDirSize](#) (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)

### 1.13.1 Detailed Description

### 1.13.2 Function Documentation

#### 1.13.2.1 int32\_t get\_pe\_section ( struct cli\_exe\_section \* section, uint32\_t num )

Gets information about the specified PE section.

##### Parameters

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

##### Returns

- 0 - success
- 1 - failure

#### 1.13.2.2 static force\_inline uint32\_t getEntryPoint ( void ) [static]

Returns the offset of the EntryPoint in the executable file.

##### Returns

offset of EP as 32-bit unsigned integer

#### 1.13.2.3 static force\_inline uint32\_t getExeOffset ( void ) [static]

Returns the offset of the executable in the file.

##### Returns

offset of embedded executable inside file

#### 1.13.2.4 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).

##### Returns

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

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

Returns the number of sections in this executable file.

Returns

number of sections as 16-bit unsigned integer

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

Return the PE BaseOfCode.

Returns

PE BaseOfCode, or 0 if not in PE hook

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

Return the PE BaseOfData.

Returns

PE BaseOfData, or 0 if not in PE hook

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

Returns PE characteristics.

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

Returns

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

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

Return the PE CheckSum.

Returns

PE CheckSum, or 0 if not in PE hook

1.13.2.10 `static force_inline uint32_t getPEDDataDirRVA ( unsigned n ) [static]`

Gets the virtual address of specified image data directory.

Parameters

in	<i>n</i>	image directory requested
----	----------	---------------------------

Returns

Virtual Address of requested image directory

1.13.2.11 `static force_inline uint32_t getPEDDataDirSize ( unsigned n ) [static]`

Gets the size of the specified image data directory.

**Parameters**

<i>in</i>	<i>n</i>	image directory requested
-----------	----------	---------------------------

**Returns**

Size of requested image directory

**1.13.2.12** `static force_inline uint16_t getPEDllCharacteristics ( void ) [static]`

Return the PE DllCharacteristics.

**Returns**

PE DllCharacteristics, or 0 if not in PE hook

**1.13.2.13** `static force_inline uint32_t getPEFileAlignment ( void ) [static]`

Return the PE FileAlignment.

**Returns**

PE FileAlignment, or 0 if not in PE hook

**1.13.2.14** `static force_inline uint64_t getPEImageBase ( void ) [static]`

Return the PE ImageBase as 64-bit integer.

**Returns**

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

**1.13.2.15** `static force_inline bool getPEisDLL ( ) [static]`

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

**Returns**

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

**1.13.2.16** `static uint32_t getPELFANew ( void ) [static]`

Gets the offset to the PE header.

**Returns**

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

**1.13.2.17** `static force_inline uint32_t getPELoaderFlags ( void ) [static]`

Return the PE LoaderFlags.

**Returns**

PE LoaderFlags or 0 if not in PE hook

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

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

Returns

PE Machine or 0 if not in PE hook

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

Return the PE MajorImageVersion.

Returns

PE MajorImageVersion, or 0 if not in PE hook

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

Returns MajorLinkerVersion for this PE file.

Returns

PE MajorLinkerVersion or 0 if not in PE hook

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

Return the PE MajorOperatingSystemVersion.

Returns

PE MajorOperatingSystemVersion, or 0 if not in PE hook

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

Return the PE MajorSubsystemVersion.

Returns

PE MajorSubsystemVersion or 0 if not in PE hook

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

Return the PE MinorImageVersion.

Returns

PE MinorImageVersion, or 0 if not in PE hook

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

Returns MinorLinkerVersion for this PE file.

Returns

PE MinorLinkerVersion or 0 if not in PE hook

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

Return the PE MinorOperatingSystemVersion.

**Returns**

PE MinorOperatingSystemVersion, or 0 if not in PE hook

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

Return the PE MinorSubsystemVersion.

**Returns**

PE MinorSubsystemVersion, or 0 if not in PE hook

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

Returns the PE number of debug symbols

**Returns**

PE NumberOfSymbols or 0 if not in PE hook

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

Returns pointer to the PE debug symbol table

**Returns**

PE PointerToSymbolTable or 0 if not in PE hook

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

Return the PE SectionAlignment.

**Returns**

PE SectionAlignment, or 0 if not in PE hook

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

Return the PE SizeOfCode.

**Returns**

PE SizeOfCode or 0 if not in PE hook

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

Return the PE SizeOfHeaders.

**Returns**

PE SizeOfHeaders, or 0 if not in PE hook

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

Return the PE SizeOfHeapCommit.

Returns

PE SizeOfHeapCommit, or 0 if not in PE hook

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

Return the PE SizeOfHeapReserve.

Returns

PE SizeOfHeapReserve, or 0 if not in PE hook

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

Return the PE SizeOfImage.

Returns

PE SizeOfImage, or 0 if not in PE hook

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

Return the PE SizeOfInitializedData.

Returns

PE SizeOfInitializeData or 0 if not in PE hook

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

Returns the size of PE optional header.

Returns

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

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

Return the PE SizeOfStackCommit.

Returns

PE SizeOfStackCommit, or 0 if not in PE hook

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

Return the PE SizeOfStackReserve.

Returns

PE SizeOfStackReserver, or 0 if not in PE hook

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

Return the PE SizeOfUninitializedData.

Returns

PE SizeOfUninitializedData or 0 if not in PE hook

**1.13.2.40** `static force_inline uint16_t getPESubsystem ( void ) [static]`

Return the PE Subsystem.

**Returns**

PE subsystem, or 0 if not in PE hook

**1.13.2.41** `static force_inline uint32_t getPETimeDateStamp ( ) [static]`

Returns the PE TimeDateStamp from headers

**Returns**

PE TimeDateStamp or 0 if not in PE hook

**1.13.2.42** `static force_inline uint32_t getPEWin32VersionValue ( void ) [static]`

Return the PE Win32VersionValue.

**Returns**

PE Win32VersionValue, or 0 if not in PE hook

**1.13.2.43** `static uint32_t getSectionRVA ( unsigned i ) [static]`

Return the RVA of the specified section.

**Parameters**

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

**Returns**

RVA of section, or -1 if invalid

**1.13.2.44** `static uint32_t getSectionVirtualSize ( unsigned i ) [static]`

Return the virtual size of the specified section.

**Parameters**

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

**Returns**

VSZ of section, or -1 if invalid

**1.13.2.45** `static uint32_t getVirtualEntryPoint ( void ) [static]`

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

**Returns**

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

**1.13.2.46** `static force_inline bool hasExeInfo ( void ) [static]`

Returns whether the current file has executable information.

**Returns**

true if the file has exe info, false otherwise

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

Returns whether PE information is available

#### Returns

true if PE information is available (in PE hooks)

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

Returns whether this is a PE32+ executable.

#### Returns

true if this is a PE32+ executable

1.13.2.49 `uint32_t pe_rawaddr ( uint32_t rva )`

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

#### Parameters

in	<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.

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

Read name of requested PE section.

#### Parameters

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

#### Returns

0 if successful,  
<0 otherwise

1.13.2.51 `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.

#### Parameters

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

#### Returns

true on success (full read)  
false on any failure



## 1.14 Scan Control

### Functions

- uint32\_t [setvirusname](#) (const uint8\_t \*name, uint32\_t len)
- int32\_t [extract\\_new](#) (int32\_t id)
- int32\_t [bytecode\\_rt\\_error](#) (int32\_t locationid)
- int32\_t [extract\\_set\\_container](#) (uint32\_t container)
- int32\_t [input\\_switch](#) (int32\_t extracted\_file)
- static force\_inline overloadable\_func void [foundVirus](#) (const char \*virusname)

#### 1.14.1 Detailed Description

#### 1.14.2 Function Documentation

##### 1.14.2.1 int32\_t [bytecode\\_rt\\_error](#) ( int32\_t *locationid* )

Report a runtime error at the specified locationID.

#### Parameters

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

#### Returns

0

##### 1.14.2.2 int32\_t [extract\\_new](#) ( int32\_t *id* )

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

#### Parameters

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

#### Returns

1 if previous extracted file was infected

##### 1.14.2.3 int32\_t [extract\\_set\\_container](#) ( uint32\_t *container* )

Sets the container type for the currently extracted file.

#### Parameters

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

#### Returns

current setting for container (CL\_TYPE\_ANY default)

##### 1.14.2.4 static force\_inline overloadable\_func void [foundVirus](#) ( const char \* *virusname* ) [static]

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

## Parameters

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

## See Also

[VIRUSNAMES](#)1.14.2.5 `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

<code>in</code>	<code>extracted_file</code>	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

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

Sets the name of the virus found.

## Parameters

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

## Returns

0

## 1.15 String Operations

### Functions

- `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 entropy_buffer` (`uint8_t *buffer`, `int32_t size`)
- static force\_inline 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 void \* `memcpy` (`void *restrict dst`, `const void *restrict src`, `uintptr_t n`) `__attribute__((nothrow)) __attribute__((__nonnull__(1)))`
- void void int `memcmp` (`const void *s1`, `const void *s2`, `uint32_t n`) `__attribute__((nothrow)) __attribute__((__pure__)) __attribute__((__nonnull__(1)))`

#### 1.15.1 Detailed Description

#### 1.15.2 Function Documentation

##### 1.15.2.1 `int32_t atoi ( const uint8_t * str, int32_t size )`

Converts string to positive number.

#### Parameters

<code>in</code>	<code>str</code>	buffer
<code>in</code>	<code>size</code>	size of <code>str</code>

#### Returns

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

##### 1.15.2.2 `uint32_t entropy_buffer ( uint8_t * buffer, int32_t size )`

Returns an approximation for the entropy of `buffer`.

#### Parameters

<code>in</code>	<code>buffer</code>	input buffer
<code>in</code>	<code>size</code>	size of buffer

#### Returns

entropy estimation \* 2<sup>26</sup>

##### 1.15.2.3 `int32_t hex2ui ( uint32_t hex1, uint32_t hex2 )`

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

#### Parameters

<code>in</code>	<code>hex1</code>	hexadecimal character
-----------------	-------------------	-----------------------

in	hex2	hexadecimal character
----	------	-----------------------

**Returns**

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

#### 1.15.2.4 static force\_inline void\* memchr ( const void \* s, int c, size\_t n ) [static]

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

**Parameters**

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

**Returns**

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

#### 1.15.2.5 void void int memcmp ( const void \* s1, const void \* s2, uint32\_t n )

[LLVM Intrinsic] Compares two memory buffers, *s1* and *s2* to length *n*.

**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*.

#### 1.15.2.6 void void\* memcpy ( void \*restrict dst, const void \*restrict src, uintptr\_t n )

[LLVM Intrinsic] Copies data between two non-overlapping buffers, from *src* to *dst* to length *n*.

**Parameters**

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

**Returns**

*dst*

#### 1.15.2.7 void\* memmove ( void \* dst, const void \* src, uintptr\_t n )

[LLVM Intrinsic] Copies data between overlapping buffers, from *src* to *dst* to length *n*.

**Parameters**

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

**Returns**

*dst*

**1.15.2.8** `void* memset ( void * src, int c, uintptr_t n )`

[LLVM Intrinsic] Fills *src* location with *c* up to length *n*.

**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*

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

Return position of match, -1 otherwise.

**Parameters**

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

**Returns**

location of match, -1 otherwise

## 2 Data Structure Documentation

### 2.1 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](#)

#### 2.1.1 Detailed Description

Executable file information.

#### 2.1.2 Field Documentation

##### 2.1.2.1 uint32\_t ep

Entrypoint of executable

##### 2.1.2.2 uint32\_t hdr\_size

Address size - PE ONLY

##### 2.1.2.3 uint16\_t nsections

Number of sections

##### 2.1.2.4 uint32\_t offset

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

##### 2.1.2.5 uint32\_t res\_addr

Resources RVA - PE ONLY

##### 2.1.2.6 struct cli\_exe\_section\* section

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

### 2.2 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](#)

### 2.2.1 Detailed Description

Section of executable file.

### 2.2.2 Field Documentation

#### 2.2.2.1 `uint32_t chr`

Section characteristics

#### 2.2.2.2 `uint32_t raw`

Raw offset (in file)

#### 2.2.2.3 `uint32_t rsz`

Raw size (in file)

#### 2.2.2.4 `uint32_t rva`

Relative VirtualAddress

#### 2.2.2.5 `uint32_t uraw`

PE - unaligned PointerToRawData

#### 2.2.2.6 `uint32_t ursz`

PE - unaligned SizeOfRawData

#### 2.2.2.7 `uint32_t urva`

PE - unaligned VirtualAddress

#### 2.2.2.8 `uint32_t uvsz`

PE - unaligned VirtualSize

#### 2.2.2.9 `uint32_t vsz`

VirtualSize

## 2.3 `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`

### 2.3.1 Detailed Description

Data for the bytecode PE hook

### 2.3.2 Field Documentation

#### 2.3.2.1 struct `pe_image_data_dir` `dirs[16]`

PE data directory header

#### 2.3.2.2 `uint32_t` `e_lfanew`

address of new exe header

#### 2.3.2.3 `uint32_t` `ep`

EntryPoint as file offset

#### 2.3.2.4 struct `pe_image_file_hdr` `file_hdr`

Header for this PE file

#### 2.3.2.5 `uint32_t` `hdr_size`

internally needed by rawaddr

#### 2.3.2.6 `uint16_t` `nsections`

Number of sections

#### 2.3.2.7 struct `pe_image_optional_hdr32` `opt32`

32-bit PE optional header

#### 2.3.2.8 struct `pe_image_optional_hdr64` `opt64`

64-bit PE optional header

#### 2.3.2.9 `uint32_t` `overlays`

number of overlays

#### 2.3.2.10 `int32_t` `overlays_sz`

size of overlays

## 2.4 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`

### 2.4.1 Detailed Description

Disassembled operand.



## 2.4.2 Field Documentation

### 2.4.2.1 enum `DIS_SIZE` `access_size`

size of access

### 2.4.2.2 enum `DIS_ACCESS` `access_type`

type of access

### 2.4.2.3 struct `DIS_mem_arg` `mem`

memory operand

### 2.4.2.4 `uint64_t` `other`

other operand

### 2.4.2.5 enum `X86REGS` `reg`

register operand

## 2.5 `DIS_fixed` Struct Reference

### Data Fields

- enum `X86OPS` `x86_opcode`
- enum `DIS_SIZE` `operation_size`
- enum `DIS_SIZE` `address_size`
- `uint8_t` `segment`
- struct `DIS_arg` `arg` [3]

### 2.5.1 Detailed Description

Disassembled instruction.

### 2.5.2 Field Documentation

#### 2.5.2.1 enum `DIS_SIZE` `address_size`

size of address

#### 2.5.2.2 struct `DIS_arg` `arg`[3]

arguments

#### 2.5.2.3 enum `DIS_SIZE` `operation_size`

size of operation

#### 2.5.2.4 `uint8_t` `segment`

segment

#### 2.5.2.5 enum `X86OPS` `x86_opcode`

opcode of X86 instruction

## 2.6 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`

### 2.6.1 Detailed Description

Disassembled memory operand: `scale_reg*scale + add_reg + displacement`.

### 2.6.2 Field Documentation

#### 2.6.2.1 enum [DIS\\_SIZE](#) `access_size`

size of access

#### 2.6.2.2 enum [X86REGS](#) `add_reg`

register used as displacement

#### 2.6.2.3 `int32_t` `displacement`

displacement as immediate number

#### 2.6.2.4 `uint8_t` `scale`

scale as immediate number

#### 2.6.2.5 enum [X86REGS](#) `scale_reg`

register used as scale

## 2.7 DISASM\_RESULT Struct Reference

### 2.7.1 Detailed Description

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

## 2.8 pe\_image\_data\_dir Struct Reference

### 2.8.1 Detailed Description

PE data directory header

## 2.9 pe\_image\_file\_hdr Struct Reference

### Data Fields

- `uint32_t` `Magic`
- `uint16_t` `Machine`
- `uint16_t` `NumberOfSections`

- uint32\_t [TimeStamp](#)
- uint32\_t [PointerToSymbolTable](#)
- uint32\_t [NumberOfSymbols](#)
- uint16\_t [SizeOfOptionalHeader](#)

### 2.9.1 Detailed Description

Header for this PE file

### 2.9.2 Field Documentation

#### 2.9.2.1 uint16\_t Machine

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

#### 2.9.2.2 uint32\_t Magic

PE magic header: PE\0\0

#### 2.9.2.3 uint16\_t NumberOfSections

Number of sections in this executable

#### 2.9.2.4 uint32\_t NumberOfSymbols

debug

#### 2.9.2.5 uint32\_t PointerToSymbolTable

debug

#### 2.9.2.6 uint16\_t SizeOfOptionalHeader

== 224

#### 2.9.2.7 uint32\_t TimeDateStamp

Unreliable

## 2.10 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](#)

### 2.10.1 Detailed Description

32-bit PE optional header

### 2.10.2 Field Documentation

#### 2.10.2.1 uint32\_t CheckSum

NT drivers only

#### 2.10.2.2 uint32\_t FileAlignment

usually 32 or 512

#### 2.10.2.3 uint32\_t ImageBase

multiple of 64 KB

#### 2.10.2.4 uint16\_t MajorImageVersion

unreliable

#### 2.10.2.5 uint8\_t MajorLinkerVersion

unreliable

#### 2.10.2.6 uint16\_t MajorOperatingSystemVersion

not used

#### 2.10.2.7 uint16\_t MinorImageVersion

unreliable

#### 2.10.2.8 uint8\_t MinorLinkerVersion

unreliable

#### 2.10.2.9 uint16\_t MinorOperatingSystemVersion

not used

#### 2.10.2.10 uint32\_t NumberOfRvaAndSizes

unreliable

#### 2.10.2.11 uint32\_t SectionAlignment

usually 32 or 4096

#### 2.10.2.12 uint32\_t SizeOfCode

unreliable

#### 2.10.2.13 uint32\_t SizeOfInitializedData

unreliable

#### 2.10.2.14 uint32\_t SizeOfUninitializedData

unreliable

## 2.11 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](#)

### 2.11.1 Detailed Description

PE 64-bit optional header

### 2.11.2 Field Documentation

#### 2.11.2.1 uint32\_t CheckSum

NT drivers only

#### 2.11.2.2 uint32\_t FileAlignment

usually 32 or 512

#### 2.11.2.3 uint64\_t ImageBase

multiple of 64 KB

#### 2.11.2.4 uint16\_t MajorImageVersion

unreliable

#### 2.11.2.5 uint8\_t MajorLinkerVersion

unreliable

#### 2.11.2.6 uint16\_t MajorOperatingSystemVersion

not used

#### 2.11.2.7 uint16\_t MinorImageVersion

unreliable

#### 2.11.2.8 uint8\_t MinorLinkerVersion

unreliable

## 2.11.2.9 uint16\_t MinorOperatingSystemVersion

not used

## 2.11.2.10 uint32\_t NumberOfRvaAndSizes

unreliable

## 2.11.2.11 uint32\_t SectionAlignment

usually 32 or 4096

## 2.11.2.12 uint32\_t SizeOfCode

unreliable

## 2.11.2.13 uint32\_t SizeOfInitializedData

unreliable

## 2.11.2.14 uint32\_t SizeOfUninitializedData

unreliable

## 2.12 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](#)

## 2.12.1 Detailed Description

PE section header

## 2.12.2 Field Documentation

## 2.12.2.1 uint8\_t Name[8]

may not end with NULL

## 2.12.2.2 uint16\_t NumberOfLinenumbers

object files only

## 2.12.2.3 uint16\_t NumberOfRelocations

object files only

## 2.12.2.4 uint32\_t PointerToLinenumbers

object files only

#### 2.12.2.5 uint32\_t PointerToRawData

offset to the section's data

#### 2.12.2.6 uint32\_t PointerToRelocations

object files only

#### 2.12.2.7 uint32\_t SizeOfRawData

multiple of FileAlignment

## 3 File Documentation

### 3.1 bytecode\_api.h File Reference

#### Enumerations

- enum `BytecodeKind` {  
`BC_GENERIC` =0, `BC_STARTUP` =1 , `BC_LOGICAL` =256, `BC_PE_UNPACKER`,  
`BC_PDF`, `BC_PE_ALL` }
- enum { `PE_INVALID_RVA` = 0xFFFFFFFF }
- enum `FunctionalityLevels` {  
`FUNC_LEVEL_096` = 51 , `FUNC_LEVEL_096_1` = 53 , `FUNC_LEVEL_096_2` = 54 , `FUNC_LEVEL_096_3`  
= 55,  
`FUNC_LEVEL_096_4` = 56, `FUNC_LEVEL_096_5` = 58, `FUNC_LEVEL_097` = 60, `FUNC_LEVEL_097_1` =  
61,  
`FUNC_LEVEL_097_2` = 62, `FUNC_LEVEL_097_3` = 63, `FUNC_LEVEL_097_4` = 64, `FUNC_LEVEL_097_5`  
= 65,  
`FUNC_LEVEL_097_6` = 67, `FUNC_LEVEL_097_7` = 68, `FUNC_LEVEL_097_8` = 69, `FUNC_LEVEL_098` =  
74,  
`FUNC_LEVEL_098_1` = 76, `FUNC_LEVEL_098_2` = 78 }
- enum `pdf_phase` { , `PDF_PHASE_PARSED`, `PDF_PHASE_POSTDUMP`, `PDF_PHASE_END`, `PDF_PHASE_PRE` }
- 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)
- int32\_t `write` (uint8\_t \*data, int32\_t size)
- int32\_t `seek` (int32\_t pos, uint32\_t whence)
- 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`)
- `const 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)
- `const uint8_t * pdf_getobj` (int32\_t objidx, uint32\_t amount)
- `int32_t pdf_getobjid` (int32\_t objidx)
- `int32_t pdf_getobjflags` (int32\_t objidx)
- `int32_t pdf_setobjflags` (int32\_t objidx, int32\_t flags)
- `int32_t pdf_get_offset` (int32\_t objidx)
- `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)
- `int32_t running_on_jit` (void)
- `int32_t get_file_reliability` (void)

## Variables

- `const uint32_t __clambc_match_counts` [64]  
*This is a low-level variable, use the Macros in `bytecode_local.h` instead to access it.*
- `const uint32_t __clambc_match_offsets` [64]  
*This is a low-level variable, use the Macros in `bytecode_local.h` instead to access it.*
- `const struct cli_pe_hook_data __clambc_pedata`
- `const uint32_t __clambc_filesize` [1]
- `const uint16_t __clambc_kind`

## 3.1.1 Enumeration Type Documentation

### 3.1.1.1 anonymous enum

#### Enumerator

**`PE_INVALID_RVA`** Invalid RVA specified

## 3.1.2 Function Documentation

### 3.1.2.1 `uint32_t test1 ( uint32_t a, uint32_t b )`

Test api.

#### Parameters

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

#### Returns

0x12345678 if parameters match, 0x55 otherwise

### 3.1.2.2 `uint32_t test2 ( uint32_t a )`

Test api2.

## Parameters

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

## Returns

0xd00f if parameter matches, 0x5555 otherwise

## 3.2 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, OPBTC, 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_CMOVBP,
  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_INVLD, 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_MOVSQ, OP_MOVSQ, OP_MOVSQ,
  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_FLDQ, OP_FLDENV, OP_FLDL2E, OP_FLDL2T,
  OP_FLDLG2, OP_FLDLN2, OP_FLDPI, OP_FLDZ,
  OP_FEMUL, OP_FEMULP, OP_FNOP, OP_FPTAN

```

- `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`

### 3.2.1 Enumeration Type Documentation

#### 3.2.1.1 enum `DIS_ACCESS`

Access type

Enumerator

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

#### 3.2.1.2 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

#### 3.2.1.3 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  
**OP\_BTC** Bit Test and Complement  
**OP\_BTR** Bit Test and Reset  
**OP\_BTS** 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\_MOVSQ** Move Data from String to String Doubleword

**OP\_MOVSQ** Move Data from String to String Word

**OP\_MOVSQ** Move Data from String to String Byte

**OP\_MOVSQ** Move with Sign-Extension

**OP\_MOVSQ** 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\_OUTQ** Output String to Port Doubleword

**OP\_OUTQ** Output String to Port Word

**OP\_OUTQ** Output String to Port Bytes

**OP\_PUSH** Push Onto the Stack

**OP\_PUSHQ** Push All Double General Purpose Registers

**OP\_PUSHQ** Push EFLAGS Register onto the Stack

**OP\_POP** Pop a Value from the Stack

**OP\_POPQ** Pop All Double General Purpose Registers from the Stack

**OP\_POPQ** Pop Stack into EFLAGS Register

**OP\_RCL** Rotate Carry Left

**OP\_RCR** Rotate Carry Right

**OP\_RDMSR** Read from Model Specific Register

**OP\_RDPMC** Read Performance Monitoring Counters

**OP\_RDTSC** Read Time-Stamp Counter

**OP\_PREFIX\_REPE** Repeat String Operation Prefix while Equal

**OP\_PREFIX\_REPN** Repeat String Operation Prefix while Not Equal

**OP\_RETF** 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\_SCASQ** Scan String Doubleword

**OP\_SCASQ** 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\_FCMOV B** Floating Point Move on Below  
**OP\_FCMOV BE** Floating Point Move on Below or Equal  
**OP\_FCMOV E** Floating Point Move on Equal  
**OP\_FCMOV NB** Floating Point Move on Not Below  
**OP\_FCMOV NBE** Floating Point Move on Not Below or Equal  
**OP\_FCMOV NE** Floating Point Move on Not Equal  
**OP\_FCMOV NU** Floating Point Move on Not Unordered  
**OP\_FCMOV U** 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  $y \cdot \log_2(x+1)$

#### 3.2.1.4 `enum X86REGS`

X86 registers

### 3.3 bytecode\_execs.h File Reference

#### Data Structures

- struct [cli\\_exe\\_section](#)
- struct [cli\\_exe\\_info](#)

### 3.4 bytecode\_local.h File Reference

#### Data Structures

- struct [DIS\\_mem\\_arg](#)
- struct [DIS\\_arg](#)
- struct [DIS\\_fixed](#)

#### Macros

- #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 [SIGNATURES\\_END](#) };
- #define [SIGNATURES\\_DEF\\_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 uint32\_t force\_inline `be32_to_host` (uint32\_t v)
- static uint64\_t force\_inline `le64_to_host` (uint64\_t v)
- static uint64\_t force\_inline `be64_to_host` (uint64\_t v)
- static uint16\_t force\_inline `le16_to_host` (uint16\_t v)
- static uint16\_t force\_inline `be16_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 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)
- 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 `getPETimestamp` ()
- 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 `getPEDataDirRVA` (unsigned n)
- static force\_inline uint32\_t `getPEDataDirSize` (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 force\_inline 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 void \* [memcpy](#) (void \*restrict dst, const void \*restrict src, uintptr\_t n) \_\_attribute\_\_((nothrow)) \_\_attribute\_\_((nonnull\_\_(1)))
- void void int [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)

### 3.4.1 Macro Definition Documentation

#### 3.4.1.1 #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

#### 3.4.1.2 #define SIGNATURES\_END ;

Old macro used to mark the end of the subsignature pattern definitions.

### 3.4.2 Function Documentation

#### 3.4.2.1 static force\_inline void overloadable\_func foundVirus ( void ) [static]

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

#### 3.4.2.2 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!

##### Parameters

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

##### Returns

$2^{26 \cdot \log_2(a/b)}$

## 3.5 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](#)

## Index

- \_\_clambc\_filesize
    - Global Variables, [27](#)
  - \_\_clambc\_kind
    - Global Variables, [27](#)
  - \_\_clambc\_match\_counts
    - Global Variables, [27](#)
  - \_\_clambc\_match\_offsets
    - Global Variables, [27](#)
  - \_\_clambc\_pedata
    - Global Variables, [27](#)
  - \_\_is\_bigendian
    - Environment, [19](#)
- ACCESS\_IMM
  - bytecode\_disasm.h, [65](#)
- ACCESS\_MEM
  - bytecode\_disasm.h, [65](#)
- ACCESS\_NOARG
  - bytecode\_disasm.h, [65](#)
- ACCESS\_REG
  - bytecode\_disasm.h, [65](#)
- ACCESS\_REL
  - bytecode\_disasm.h, [65](#)
- Abstract Data Types, [1](#)
  - buffer\_pipe\_done, [2](#)
  - buffer\_pipe\_new, [2](#)
  - buffer\_pipe\_new\_fromfile, [2](#)
  - buffer\_pipe\_read\_avail, [2](#)
  - buffer\_pipe\_read\_get, [3](#)
  - buffer\_pipe\_read\_stopped, [3](#)
  - buffer\_pipe\_write\_avail, [3](#)
  - buffer\_pipe\_write\_get, [3](#)
  - buffer\_pipe\_write\_stopped, [3](#)
  - hashset\_add, [4](#)
  - hashset\_contains, [4](#)
  - hashset\_done, [4](#)
  - hashset\_empty, [4](#)
  - hashset\_new, [5](#)
  - hashset\_remove, [5](#)
  - inflate\_done, [5](#)
  - inflate\_init, [5](#)
  - inflate\_process, [5](#)
  - malloc, [6](#)
  - map\_addkey, [6](#)
  - map\_done, [6](#)
  - map\_find, [6](#)
  - map\_getvalue, [7](#)
  - map\_getvaluesize, [7](#)
  - map\_new, [7](#)
  - map\_remove, [7](#)
  - map\_setvalue, [8](#)
- access\_size
  - DIS\_arg, [54](#)
  - DIS\_mem\_arg, [55](#)
- access\_type
  - DIS\_arg, [54](#)
- add\_reg
  - DIS\_mem\_arg, [55](#)
- address\_size
  - DIS\_fixed, [54](#)
- arg
  - DIS\_fixed, [54](#)
- atoi
  - String Operations, [48](#)
- BC\_GENERIC
  - Bytecode Configuration, [11](#)
- BC\_LOGICAL
  - Bytecode Configuration, [11](#)
- BC\_PDF
  - Bytecode Configuration, [11](#)
- BC\_PE\_ALL
  - Bytecode Configuration, [11](#)
- BC\_PE\_UNPACKER
  - Bytecode Configuration, [11](#)
- BC\_STARTUP
  - Bytecode Configuration, [11](#)
- be16\_to\_host
  - Environment, [19](#)
- be32\_to\_host
  - Environment, [19](#)
- be64\_to\_host
  - Environment, [20](#)
- buffer\_pipe\_done
  - Abstract Data Types, [2](#)
- buffer\_pipe\_new
  - Abstract Data Types, [2](#)
- buffer\_pipe\_new\_fromfile
  - Abstract Data Types, [2](#)
- buffer\_pipe\_read\_avail
  - Abstract Data Types, [2](#)
- buffer\_pipe\_read\_get
  - Abstract Data Types, [3](#)
- buffer\_pipe\_read\_stopped
  - Abstract Data Types, [3](#)
- buffer\_pipe\_write\_avail
  - Abstract Data Types, [3](#)
- buffer\_pipe\_write\_get
  - Abstract Data Types, [3](#)
- buffer\_pipe\_write\_stopped
  - Abstract Data Types, [3](#)
- Bytecode Configuration, [9](#)
  - BC\_GENERIC, [11](#)
  - BC\_LOGICAL, [11](#)
  - BC\_PDF, [11](#)
  - BC\_PE\_ALL, [11](#)
  - BC\_PE\_UNPACKER, [11](#)
  - BC\_STARTUP, [11](#)
  - BytecodeKind, [11](#)
  - COPYRIGHT, [9](#)

DECLARE\_SIGNATURE, 9  
 FUNC\_LEVEL\_096, 11  
 FUNC\_LEVEL\_096\_1, 11  
 FUNC\_LEVEL\_096\_2, 11  
 FUNC\_LEVEL\_096\_3, 11  
 FUNC\_LEVEL\_096\_4, 11  
 FUNC\_LEVEL\_096\_5, 11  
 FUNC\_LEVEL\_097, 12  
 FUNC\_LEVEL\_097\_1, 12  
 FUNC\_LEVEL\_097\_2, 12  
 FUNC\_LEVEL\_097\_3, 12  
 FUNC\_LEVEL\_097\_4, 12  
 FUNC\_LEVEL\_097\_5, 12  
 FUNC\_LEVEL\_097\_6, 12  
 FUNC\_LEVEL\_097\_7, 12  
 FUNC\_LEVEL\_097\_8, 12  
 FUNC\_LEVEL\_098, 12  
 FUNC\_LEVEL\_098\_1, 12  
 FUNC\_LEVEL\_098\_2, 12  
 FunctionalityLevels, 11  
 ICONGROUP1, 10  
 ICONGROUP2, 10  
 PDF\_HOOK\_DECLARE, 10  
 PE\_HOOK\_DECLARE, 10  
 PE\_UNPACKER\_DECLARE, 10  
 SIGNATURES\_DECL\_END, 10  
 SIGNATURES\_DEF\_END, 10  
 TARGET, 11  
 VIRUSNAME\_PREFIX, 11  
 VIRUSNAMES, 11  
 bytecode\_api.h  
   PE\_INVALID\_RVA, 62  
 bytecode\_disasm.h  
   ACCESS\_IMM, 65  
   ACCESS\_MEM, 65  
   ACCESS\_NOARG, 65  
   ACCESS\_REG, 65  
   ACCESS\_REL, 65  
   OP\_AAA, 65  
   OP\_AAD, 65  
   OP\_AAM, 65  
   OP\_AAS, 65  
   OP\_ADC, 65  
   OP\_ADD, 65  
   OP\_AND, 65  
   OP\_ARPL, 65  
   OP\_BOUND, 65  
   OP\_BSF, 65  
   OP\_BSR, 65  
   OP\_BSWAP, 66  
   OP\_BT, 66  
   OPBTC, 66  
   OPBTR, 66  
   OPBTS, 66  
   OP\_CALL, 66  
   OP\_CBW, 66  
   OP\_CDQ, 66  
   OP\_CLC, 66  
   OP\_CLD, 66  
   OP\_CLI, 66  
   OP\_CLTS, 66  
   OP\_CMC, 66  
   OP\_CMOVA, 66  
   OP\_CMOVBE, 66  
   OP\_CMOVC, 66  
   OP\_CMOVG, 66  
   OP\_CMOVGE, 66  
   OP\_CMOVL, 66  
   OP\_CMOVLE, 66  
   OP\_CMOVNC, 66  
   OP\_CMOVNO, 66  
   OP\_CMOVNP, 66  
   OP\_CMOVNS, 66  
   OP\_CMOVNZ, 66  
   OP\_CMOVO, 66  
   OP\_CMOVP, 66  
   OP\_CMOVS, 66  
   OP\_CMOVZ, 66  
   OP\_CMP, 66  
   OP\_CMPSB, 66  
   OP\_CMPSD, 66  
   OP\_CMPSW, 66  
   OP\_CMPXCHG, 66  
   OP\_CMPXCHG8B, 66  
   OP\_CPUID, 66  
   OP\_CWDE, 66  
   OP\_DAA, 66  
   OP\_DAS, 66  
   OP\_DEC, 66  
   OP\_DIV, 66  
   OP\_ENTER, 66  
   OP\_F2XM1, 70  
   OP\_FABS, 70  
   OP\_FADD, 70  
   OP\_FADDP, 70  
   OP\_FBLD, 70  
   OP\_FBSTP, 70  
   OP\_FCHS, 70  
   OP\_FCLEX, 70  
   OP\_FCMOVB, 70  
   OP\_FCMOVBE, 70  
   OP\_FCMOVE, 70  
   OP\_FCMOVNB, 70  
   OP\_FCMOVNBE, 70  
   OP\_FCMOVNE, 70  
   OP\_FCMOVNU, 70  
   OP\_FCMOVU, 70  
   OP\_FCOM, 70  
   OP\_FCOMI, 70  
   OP\_FCOMIP, 70  
   OP\_FCOMP, 70  
   OP\_FCOMPP, 70  
   OP\_FCOS, 70  
   OP\_FDECSTP, 70  
   OP\_FDIV, 70  
   OP\_FDIVP, 70

OP\_FDIVR, 70  
OP\_FDIVRP, 70  
OP\_FFREE, 70  
OP\_FIADD, 70  
OP\_FICOM, 70  
OP\_FICOMP, 70  
OP\_FIDIV, 70  
OP\_FIDIVR, 70  
OP\_FILD, 70  
OP\_FIMUL, 70  
OP\_FINCSTP, 70  
OP\_FINIT, 70  
OP\_FIST, 70  
OP\_FISTP, 70  
OP\_FISTTP, 70  
OP\_FISUB, 70  
OP\_FISUBR, 70  
OP\_FLD, 70  
OP\_FLD1, 70  
OP\_FLDCW, 71  
OP\_FLDENV, 71  
OP\_FLDL2E, 71  
OP\_FLDL2T, 71  
OP\_FLDLG2, 71  
OP\_FLDLN2, 71  
OP\_FLDPI, 71  
OP\_FLDZ, 71  
OP\_FMUL, 71  
OP\_FMULP, 71  
OP\_FNOP, 71  
OP\_FPATAN, 71  
OP\_FPREM, 71  
OP\_FPREM1, 71  
OP\_FPTAN, 71  
OP\_FPU, 70  
OP\_FRNDINT, 71  
OP\_FRSTOR, 71  
OP\_FSAVE, 71  
OP\_FSCALE, 71  
OP\_FSINCOS, 71  
OP\_FSQRT, 71  
OP\_FST, 71  
OP\_FSTCW, 71  
OP\_FSTENV, 71  
OP\_FSTP, 71  
OP\_FSTSW, 71  
OP\_FSUB, 71  
OP\_FSUBP, 71  
OP\_FSUBR, 71  
OP\_FSUBRP, 71  
OP\_FTST, 71  
OP\_FUCOM, 71  
OP\_FUCOMI, 71  
OP\_FUCOMIP, 71  
OP\_FUCOMP, 71  
OP\_FUCOMPP, 71  
OP\_FWAIT, 66  
OP\_FXAM, 71  
OP\_FXCH, 71  
OP\_FXTRACT, 71  
OP\_FYL2X, 71  
OP\_FYL2XP1, 71  
OP\_HLT, 66  
OP\_IDIV, 66  
OP\_IMUL, 67  
OP\_IN, 67  
OP\_INC, 67  
OP\_INSB, 67  
OP\_INSD, 67  
OP\_INSW, 67  
OP\_INT, 67  
OP\_INT3, 67  
OP\_INT0, 67  
OP\_INV, 67  
OP\_INVLPG, 67  
OP\_IRET, 67  
OP\_JA, 67  
OP\_JBE, 67  
OP\_JC, 67  
OP\_JECXZ, 68  
OP\_JG, 67  
OP\_JGE, 67  
OP\_JL, 67  
OP\_JLE, 67  
OP\_JMP, 67  
OP\_JNC, 67  
OP\_JNO, 67  
OP\_JNP, 67  
OP\_JNS, 67  
OP\_JNZ, 67  
OP\_JO, 67  
OP\_JP, 67  
OP\_JS, 67  
OP\_JZ, 67  
OP\_LAHF, 67  
OP\_LAR, 67  
OP\_LDS, 67  
OP\_LEA, 67  
OP\_LEAVE, 67  
OP\_LES, 67  
OP\_LFS, 67  
OP\_LGDT, 67  
OP\_LGS, 67  
OP\_LIDT, 67  
OP\_LLD, 67  
OP\_LODSB, 67  
OP\_LODSD, 67  
OP\_LODSW, 67  
OP\_LOOP, 67  
OP\_LOOPE, 68  
OP\_LOOPNE, 68  
OP\_LSL, 68  
OP\_LSS, 68  
OP\_LTR, 68  
OP\_MOV, 68  
OP\_MOVSB, 68



OP\_MOVSD, 68  
 OP\_MOVSX, 68  
 OP\_MOVZX, 68  
 OP\_MUL, 68  
 OP\_NEG, 68  
 OP\_NOP, 68  
 OP\_NOT, 68  
 OP\_OR, 68  
 OP\_OUT, 68  
 OP\_OUTSB, 68  
 OP\_OUTSD, 68  
 OP\_OUTSW, 68  
 OP\_POP, 68  
 OP\_POPAD, 68  
 OP\_POPFD, 68  
 OP\_PREFIX\_LOCK, 67  
 OP\_PREFIX\_REPE, 68  
 OP\_PREFIX\_REPN, 68  
 OP\_PUSH, 68  
 OP\_PUSHAD, 68  
 OP\_PUSHFD, 68  
 OP\_RCL, 68  
 OP\_RCR, 68  
 OP\_RDMSR, 68  
 OP\_RDPMSR, 68  
 OP\_RDTSC, 68  
 OP\_RETF, 68  
 OP\_RETN, 68  
 OP\_ROL, 68  
 OP\_ROR, 68  
 OP\_RSM, 68  
 OP\_SAHF, 68  
 OP\_SAR, 68  
 OP\_SBB, 68  
 OP\_SCASB, 68  
 OP\_SCASD, 68  
 OP\_SCASW, 68  
 OP\_SETA, 69  
 OP\_SETBE, 69  
 OP\_SETC, 69  
 OP\_SETG, 69  
 OP\_SETGE, 69  
 OP\_SETL, 69  
 OP\_SETLE, 69  
 OP\_SETNC, 69  
 OP\_SETNO, 69  
 OP\_SETNP, 69  
 OP\_SETNS, 69  
 OP\_SETNZ, 69  
 OP\_SETO, 69  
 OP\_SETP, 69  
 OP\_SETS, 69  
 OP\_SETZ, 69  
 OP\_SGDT, 69  
 OP\_SHL, 69  
 OP\_SHLD, 69  
 OP\_SHR, 69  
 OP\_SHRD, 69  
 OP\_SIDT, 69  
 OP\_SLDT, 69  
 OP\_STC, 69  
 OP\_STD, 69  
 OP\_STI, 69  
 OP\_STOSB, 69  
 OP\_STOSD, 69  
 OP\_STOSW, 69  
 OP\_STR, 69  
 OP\_SUB, 69  
 OP\_SYSCALL, 69  
 OP\_SYSENTER, 69  
 OP\_SYSEXIT, 69  
 OP\_SYSRET, 69  
 OP\_TEST, 69  
 OP\_UD2, 69  
 OP\_VERR, 69  
 OP\_VERRW, 69  
 OP\_WBINVD, 69  
 OP\_WRMSR, 69  
 OP\_XADD, 69  
 OP\_XCHG, 69  
 OP\_XLAT, 69  
 OP\_XOR, 69  
 SIZEB, 65  
 SIZED, 65  
 SIZEF, 65  
 SIZEPTR, 65  
 SIZEQ, 65  
 SIZET, 65  
 SIZEW, 65  
 bytecode\_api.h, 60  
     test1, 62  
     test2, 62  
 bytecode\_disasm.h, 63  
     DIS\_ACCESS, 65  
     DIS\_SIZE, 65  
     X86OPS, 65  
     X86REGS, 71  
 bytecode\_execs.h, 72  
 bytecode\_local.h, 72  
     foundVirus, 74  
     ilog2\_compat, 74  
     SIGNATURES\_END, 74  
 bytecode\_pe.h, 74  
 bytecode\_rt\_error  
     Scan Control, 46  
 BytecodeKind  
     Bytecode Configuration, 11  
 COPYRIGHT  
     Bytecode Configuration, 9  
 check\_platform  
     Environment, 20  
 CheckSum  
     pe\_image\_optional\_hdr32, 57  
     pe\_image\_optional\_hdr64, 58  
 chr

- cli\_exe\_section, 52
- cli\_exe\_info, 51
  - ep, 51
  - hdr\_size, 51
  - nsections, 51
  - offset, 51
  - res\_addr, 51
  - section, 51
- cli\_exe\_section, 51
  - chr, 52
  - raw, 52
  - rsz, 52
  - rva, 52
  - uraw, 52
  - ursz, 52
  - urva, 52
  - uvsz, 52
  - vsz, 52
- cli\_pe\_hook\_data, 52
  - dirs, 53
  - e\_lfanew, 53
  - ep, 53
  - file\_hdr, 53
  - hdr\_size, 53
  - nsections, 53
  - opt32, 53
  - opt64, 53
  - overlays, 53
  - overlays\_sz, 53
- cli\_readint16
  - Environment, 20
- cli\_readint32
  - Environment, 20
- cli\_writeint32
  - Environment, 20
- count\_match
  - Engine Queries, 17
- DECLARE\_SIGNATURE
  - Bytecode Configuration, 9
- DIS\_ACCESS
  - bytecode\_disasm.h, 65
- DIS\_SIZE
  - bytecode\_disasm.h, 65
- DIS\_arg, 53
  - access\_size, 54
  - access\_type, 54
  - mem, 54
  - other, 54
  - reg, 54
- DIS\_fixed, 54
  - address\_size, 54
  - arg, 54
  - operation\_size, 54
  - segment, 54
  - x86\_opcode, 54
- DIS\_mem\_arg, 55
  - access\_size, 55
  - add\_reg, 55
  - displacement, 55
  - scale, 55
  - scale\_reg, 55
- DISASM\_RESULT, 55
- debug
  - Debugging, 13
- debug\_print\_str
  - Debugging, 13
- debug\_print\_str\_nonl
  - Debugging, 15
- debug\_print\_str\_start
  - Debugging, 15
- debug\_print\_uint
  - Debugging, 15
- Debugging, 13
  - debug, 13
  - debug\_print\_str, 13
  - debug\_print\_str\_nonl, 15
  - debug\_print\_str\_start, 15
  - debug\_print\_uint, 15
- dirs
  - cli\_pe\_hook\_data, 53
- disable\_bytecode\_if
  - Environment, 21
- disable\_jit\_if
  - Environment, 21
- disasm\_x86
  - Disassembly, 16
- DisassembleAt
  - Disassembly, 16
- Disassembly, 16
  - disasm\_x86, 16
  - DisassembleAt, 16
- displacement
  - DIS\_mem\_arg, 55
- e\_lfanew
  - cli\_pe\_hook\_data, 53
- Engine Queries, 17
  - count\_match, 17
  - engine\_db\_options, 17
  - engine\_dconf\_level, 17
  - engine\_functionality\_level, 17
  - engine\_scan\_options, 17
  - match\_location, 18
  - match\_location\_check, 18
  - matches, 18
  - running\_on\_jit, 18
- engine\_db\_options
  - Engine Queries, 17
- engine\_dconf\_level
  - Engine Queries, 17
- engine\_functionality\_level
  - Engine Queries, 17
- engine\_scan\_options
  - Engine Queries, 17
- entropy\_buffer
  - String Operations, 48
- Environment, 19

- `__is_bigendian`, 19
  - `be16_to_host`, 19
  - `be32_to_host`, 19
  - `be64_to_host`, 20
  - `check_platform`, 20
  - `cli_readint16`, 20
  - `cli_readint32`, 20
  - `cli_writeint32`, 20
  - `disable_bytecode_if`, 21
  - `disable_jit_if`, 21
  - `get_environment`, 21
  - `le16_to_host`, 21
  - `le32_to_host`, 22
  - `le64_to_host`, 22
  - `version_compare`, 22
- `ep`
  - `cli_exe_info`, 51
  - `cli_pe_hook_data`, 53
- `extract_new`
  - Scan Control, 46
- `extract_set_container`
  - Scan Control, 46
- `FUNC_LEVEL_096`
  - Bytecode Configuration, 11
- `FUNC_LEVEL_096_1`
  - Bytecode Configuration, 11
- `FUNC_LEVEL_096_2`
  - Bytecode Configuration, 11
- `FUNC_LEVEL_096_3`
  - Bytecode Configuration, 11
- `FUNC_LEVEL_096_4`
  - Bytecode Configuration, 11
- `FUNC_LEVEL_096_5`
  - Bytecode Configuration, 11
- `FUNC_LEVEL_097`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_097_1`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_097_2`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_097_3`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_097_4`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_097_5`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_097_6`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_097_7`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_097_8`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_098`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_098_1`
  - Bytecode Configuration, 12
- `FUNC_LEVEL_098_2`
  - Bytecode Configuration, 12
- File Operations, 23
  - `file_byteat`, 23
  - `file_find`, 23
  - `file_find_limit`, 23
  - `fill_buffer`, 25
  - `get_file_reliability`, 25
  - `getFileSize`, 25
  - `read`, 25
  - `read_number`, 26
  - `SEEK_CUR`, 23
  - `SEEK_END`, 23
  - `SEEK_SET`, 23
  - `seek`, 26
  - `write`, 26
- `file_byteat`
  - File Operations, 23
- `file_find`
  - File Operations, 23
- `file_find_limit`
  - File Operations, 23
- `file_hdr`
  - `cli_pe_hook_data`, 53
- FileAlignment
  - `pe_image_optional_hdr32`, 57
  - `pe_image_optional_hdr64`, 58
- `fill_buffer`
  - File Operations, 25
- `foundVirus`
  - `bytecode_local.h`, 74
  - Scan Control, 46
- FunctionalityLevels
  - Bytecode Configuration, 11
- `get_environment`
  - Environment, 21
- `get_file_reliability`
  - File Operations, 25
- `get_pe_section`
  - PE Operations, 38
- `getEntryPoint`
  - PE Operations, 38
- `getExeOffset`
  - PE Operations, 38
- `getFileSize`
  - File Operations, 25
- `getImageBase`
  - PE Operations, 38
- `getNumberOfSections`
  - PE Operations, 38
- `getPEBaseOfCode`
  - PE Operations, 39
- `getPEBaseOfData`
  - PE Operations, 39
- `getPECharacteristics`
  - PE Operations, 39
- `getPEChecksum`
  - PE Operations, 39
- `getPEDataDirRVA`
  - PE Operations, 39

- getPEDataDirSize
  - PE Operations, [39](#)
- getPEDllCharacteristics
  - PE Operations, [40](#)
- getPEFileAlignment
  - PE Operations, [40](#)
- getPEImageBase
  - PE Operations, [40](#)
- getPELFANew
  - PE Operations, [40](#)
- getPELoaderFlags
  - PE Operations, [40](#)
- getPEMachine
  - PE Operations, [40](#)
- getPEMajorImageVersion
  - PE Operations, [41](#)
- getPEMajorLinkerVersion
  - PE Operations, [41](#)
- getPEMajorOperatingSystemVersion
  - PE Operations, [41](#)
- getPEMajorSubsystemVersion
  - PE Operations, [41](#)
- getPEMinorImageVersion
  - PE Operations, [41](#)
- getPEMinorLinkerVersion
  - PE Operations, [41](#)
- getPEMinorOperatingSystemVersion
  - PE Operations, [41](#)
- getPEMinorSubsystemVersion
  - PE Operations, [42](#)
- getPENumberOfSymbols
  - PE Operations, [42](#)
- getPEPointerToSymbolTable
  - PE Operations, [42](#)
- getPESectionAlignment
  - PE Operations, [42](#)
- getPESizeOfCode
  - PE Operations, [42](#)
- getPESizeOfHeaders
  - PE Operations, [42](#)
- getPESizeOfHeapCommit
  - PE Operations, [42](#)
- getPESizeOfHeapReserve
  - PE Operations, [43](#)
- getPESizeOfImage
  - PE Operations, [43](#)
- getPESizeOfInitializedData
  - PE Operations, [43](#)
- getPESizeOfOptionalHeader
  - PE Operations, [43](#)
- getPESizeOfStackCommit
  - PE Operations, [43](#)
- getPESizeOfStackReserve
  - PE Operations, [43](#)
- getPESizeOfUninitializedData
  - PE Operations, [43](#)
- getPESubsystem
  - PE Operations, [43](#)
- getPETimeDateStamp
  - PE Operations, [44](#)
- getPEWin32VersionValue
  - PE Operations, [44](#)
- getPEisDLL
  - PE Operations, [40](#)
- getSectionRVA
  - PE Operations, [44](#)
- getSectionVirtualSize
  - PE Operations, [44](#)
- getVirtualEntryPoint
  - PE Operations, [44](#)
- Global Variables, [27](#)
  - \_\_clambc\_filesize, [27](#)
  - \_\_clambc\_kind, [27](#)
  - \_\_clambc\_match\_counts, [27](#)
  - \_\_clambc\_match\_offsets, [27](#)
  - \_\_clambc\_pedata, [27](#)
- hasExeInfo
  - PE Operations, [44](#)
- hasPEInfo
  - PE Operations, [44](#)
- hashset\_add
  - Abstract Data Types, [4](#)
- hashset\_contains
  - Abstract Data Types, [4](#)
- hashset\_done
  - Abstract Data Types, [4](#)
- hashset\_empty
  - Abstract Data Types, [4](#)
- hashset\_new
  - Abstract Data Types, [5](#)
- hashset\_remove
  - Abstract Data Types, [5](#)
- hdr\_size
  - cli\_exe\_info, [51](#)
  - cli\_pe\_hook\_data, [53](#)
- hex2ui
  - String Operations, [48](#)
- ICONGROUP1
  - Bytecode Configuration, [10](#)
- ICONGROUP2
  - Bytecode Configuration, [10](#)
- Icon Matcher, [29](#)
  - matchicon, [29](#)
- icos
  - Math Operation, [30](#)
- iexp
  - Math Operation, [30](#)
- ilog2
  - Math Operation, [30](#)
- ilog2\_compat
  - bytecode\_local.h, [74](#)
- ImageBase
  - pe\_image\_optional\_hdr32, [57](#)
  - pe\_image\_optional\_hdr64, [58](#)
- inflate\_done

- Abstract Data Types, 5
- inflate\_init
  - Abstract Data Types, 5
- inflate\_process
  - Abstract Data Types, 5
- input\_switch
  - Scan Control, 47
- ipow
  - Math Operation, 30
- isPE64
  - PE Operations, 45
- isin
  - Math Operation, 31
- JavaScript Normalization, 28
  - jsnorm\_done, 28
  - jsnorm\_init, 28
  - jsnorm\_process, 28
- jsnorm\_done
  - JavaScript Normalization, 28
- jsnorm\_init
  - JavaScript Normalization, 28
- jsnorm\_process
  - JavaScript Normalization, 28
- le16\_to\_host
  - Environment, 21
- le32\_to\_host
  - Environment, 22
- le64\_to\_host
  - Environment, 22
- Machine
  - pe\_image\_file\_hdr, 56
- Magic
  - pe\_image\_file\_hdr, 56
- MajorImageVersion
  - pe\_image\_optional\_hdr32, 57
  - pe\_image\_optional\_hdr64, 58
- MajorLinkerVersion
  - pe\_image\_optional\_hdr32, 57
  - pe\_image\_optional\_hdr64, 58
- MajorOperatingSystemVersion
  - pe\_image\_optional\_hdr32, 57
  - pe\_image\_optional\_hdr64, 58
- malloc
  - Abstract Data Types, 6
- map\_addkey
  - Abstract Data Types, 6
- map\_done
  - Abstract Data Types, 6
- map\_find
  - Abstract Data Types, 6
- map\_getvalue
  - Abstract Data Types, 7
- map\_getvaluesize
  - Abstract Data Types, 7
- map\_new
  - Abstract Data Types, 7
- map\_remove
  - Abstract Data Types, 7
- map\_setvalue
  - Abstract Data Types, 8
- match\_location
  - Engine Queries, 18
- match\_location\_check
  - Engine Queries, 18
- matches
  - Engine Queries, 18
- matchicon
  - Icon Matcher, 29
- Math Operation, 30
  - icos, 30
  - iexp, 30
  - ilog2, 30
  - ipow, 30
  - isin, 31
- mem
  - DIS\_arg, 54
- memchr
  - String Operations, 49
- memcmp
  - String Operations, 49
- memcpy
  - String Operations, 49
- memmove
  - String Operations, 49
- memset
  - String Operations, 50
- memstr
  - String Operations, 50
- MinorImageVersion
  - pe\_image\_optional\_hdr32, 57
  - pe\_image\_optional\_hdr64, 58
- MinorLinkerVersion
  - pe\_image\_optional\_hdr32, 57
  - pe\_image\_optional\_hdr64, 58
- MinorOperatingSystemVersion
  - pe\_image\_optional\_hdr32, 57
  - pe\_image\_optional\_hdr64, 58
- Name
  - pe\_image\_section\_hdr, 59
- nsections
  - cli\_exe\_info, 51
  - cli\_pe\_hook\_data, 53
- NumberOfLinenumbers
  - pe\_image\_section\_hdr, 59
- NumberOfRelocations
  - pe\_image\_section\_hdr, 59
- NumberOfRvaAndSizes
  - pe\_image\_optional\_hdr32, 57
  - pe\_image\_optional\_hdr64, 59
- NumberOfSections
  - pe\_image\_file\_hdr, 56
- NumberOfSymbols
  - pe\_image\_file\_hdr, 56

OP\_AAA  
    bytecode\_disasm.h, 65

OP\_AAD  
    bytecode\_disasm.h, 65

OP\_AAM  
    bytecode\_disasm.h, 65

OP\_AAS  
    bytecode\_disasm.h, 65

OP\_ADC  
    bytecode\_disasm.h, 65

OP\_ADD  
    bytecode\_disasm.h, 65

OP\_AND  
    bytecode\_disasm.h, 65

OP\_ARPL  
    bytecode\_disasm.h, 65

OP\_BOUND  
    bytecode\_disasm.h, 65

OP\_BSF  
    bytecode\_disasm.h, 65

OP\_BSR  
    bytecode\_disasm.h, 65

OP\_BSWAP  
    bytecode\_disasm.h, 66

OP\_BT  
    bytecode\_disasm.h, 66

OP\_BTC  
    bytecode\_disasm.h, 66

OP\_BTR  
    bytecode\_disasm.h, 66

OP\_BTS  
    bytecode\_disasm.h, 66

OP\_CALL  
    bytecode\_disasm.h, 66

OP\_CBW  
    bytecode\_disasm.h, 66

OP\_CDQ  
    bytecode\_disasm.h, 66

OP\_CLC  
    bytecode\_disasm.h, 66

OP\_CLD  
    bytecode\_disasm.h, 66

OP\_CLI  
    bytecode\_disasm.h, 66

OP\_CLTS  
    bytecode\_disasm.h, 66

OP\_CMC  
    bytecode\_disasm.h, 66

OP\_CMOVA  
    bytecode\_disasm.h, 66

OP\_CMOVBE  
    bytecode\_disasm.h, 66

OP\_CMOVC  
    bytecode\_disasm.h, 66

OP\_CMOVG  
    bytecode\_disasm.h, 66

OP\_CMOVGE  
    bytecode\_disasm.h, 66

OP\_CMOVL  
    bytecode\_disasm.h, 66

OP\_CMOVLE  
    bytecode\_disasm.h, 66

OP\_CMOVNC  
    bytecode\_disasm.h, 66

OP\_CMOVNO  
    bytecode\_disasm.h, 66

OP\_CMOVNP  
    bytecode\_disasm.h, 66

OP\_CMOVNS  
    bytecode\_disasm.h, 66

OP\_CMOVNZ  
    bytecode\_disasm.h, 66

OP\_CMOVO  
    bytecode\_disasm.h, 66

OP\_CMOVP  
    bytecode\_disasm.h, 66

OP\_CMOVVS  
    bytecode\_disasm.h, 66

OP\_CMOVZ  
    bytecode\_disasm.h, 66

OP\_CMP  
    bytecode\_disasm.h, 66

OP\_CMPSB  
    bytecode\_disasm.h, 66

OP\_CMPSD  
    bytecode\_disasm.h, 66

OP\_CMPSW  
    bytecode\_disasm.h, 66

OP\_CMPXCHG  
    bytecode\_disasm.h, 66

OP\_CMPXCHG8B  
    bytecode\_disasm.h, 66

OP\_CPUID  
    bytecode\_disasm.h, 66

OP\_CWDE  
    bytecode\_disasm.h, 66

OP\_DAA  
    bytecode\_disasm.h, 66

OP\_DAS  
    bytecode\_disasm.h, 66

OP\_DEC  
    bytecode\_disasm.h, 66

OP\_DIV  
    bytecode\_disasm.h, 66

OP\_ENTER  
    bytecode\_disasm.h, 66

OP\_F2XM1  
    bytecode\_disasm.h, 70

OP\_FABS  
    bytecode\_disasm.h, 70

OP\_FADD  
    bytecode\_disasm.h, 70

OP\_FADDP  
    bytecode\_disasm.h, 70

OP\_FBLD  
    bytecode\_disasm.h, 70



OP\_FSCALE  
    bytecode\_disasm.h, 71

OP\_FSINCOS  
    bytecode\_disasm.h, 71

OP\_FSQRT  
    bytecode\_disasm.h, 71

OP\_FST  
    bytecode\_disasm.h, 71

OP\_FSTCW  
    bytecode\_disasm.h, 71

OP\_FSTENV  
    bytecode\_disasm.h, 71

OP\_FSTP  
    bytecode\_disasm.h, 71

OP\_FSTSW  
    bytecode\_disasm.h, 71

OP\_FSUB  
    bytecode\_disasm.h, 71

OP\_FSUBP  
    bytecode\_disasm.h, 71

OP\_FSUBR  
    bytecode\_disasm.h, 71

OP\_FSUBRP  
    bytecode\_disasm.h, 71

OP\_FTST  
    bytecode\_disasm.h, 71

OP\_FUCOM  
    bytecode\_disasm.h, 71

OP\_FUCOMI  
    bytecode\_disasm.h, 71

OP\_FUCOMIP  
    bytecode\_disasm.h, 71

OP\_FUCOMP  
    bytecode\_disasm.h, 71

OP\_FUCOMPP  
    bytecode\_disasm.h, 71

OP\_FWAIT  
    bytecode\_disasm.h, 66

OP\_FXAM  
    bytecode\_disasm.h, 71

OP\_FXCH  
    bytecode\_disasm.h, 71

OP\_FXTRACT  
    bytecode\_disasm.h, 71

OP\_FYL2X  
    bytecode\_disasm.h, 71

OP\_FYL2XP1  
    bytecode\_disasm.h, 71

OP\_HLT  
    bytecode\_disasm.h, 66

OP\_IDIV  
    bytecode\_disasm.h, 66

OP\_IMUL  
    bytecode\_disasm.h, 67

OP\_IN  
    bytecode\_disasm.h, 67

OP\_INC  
    bytecode\_disasm.h, 67

OP\_INSB  
    bytecode\_disasm.h, 67

OP\_INSD  
    bytecode\_disasm.h, 67

OP\_INSW  
    bytecode\_disasm.h, 67

OP\_INT  
    bytecode\_disasm.h, 67

OP\_INT3  
    bytecode\_disasm.h, 67

OP\_INT0  
    bytecode\_disasm.h, 67

OP\_INVD  
    bytecode\_disasm.h, 67

OP\_INVLPG  
    bytecode\_disasm.h, 67

OP\_IRET  
    bytecode\_disasm.h, 67

OP\_JA  
    bytecode\_disasm.h, 67

OP\_JBE  
    bytecode\_disasm.h, 67

OP\_JC  
    bytecode\_disasm.h, 67

OP\_JECXZ  
    bytecode\_disasm.h, 68

OP\_JG  
    bytecode\_disasm.h, 67

OP\_JGE  
    bytecode\_disasm.h, 67

OP\_JL  
    bytecode\_disasm.h, 67

OP\_JLE  
    bytecode\_disasm.h, 67

OP\_JMP  
    bytecode\_disasm.h, 67

OP\_JNC  
    bytecode\_disasm.h, 67

OP\_JNO  
    bytecode\_disasm.h, 67

OP\_JNP  
    bytecode\_disasm.h, 67

OP\_JNS  
    bytecode\_disasm.h, 67

OP\_JNZ  
    bytecode\_disasm.h, 67

OP\_JO  
    bytecode\_disasm.h, 67

OP\_JP  
    bytecode\_disasm.h, 67

OP\_JS  
    bytecode\_disasm.h, 67

OP\_JZ  
    bytecode\_disasm.h, 67

OP\_LAHF  
    bytecode\_disasm.h, 67

OP\_LAR  
    bytecode\_disasm.h, 67



OP\_LDS  
    bytecode\_disasm.h, [67](#)

OP\_LEA  
    bytecode\_disasm.h, [67](#)

OP\_LEAVE  
    bytecode\_disasm.h, [67](#)

OP\_LES  
    bytecode\_disasm.h, [67](#)

OP\_LFS  
    bytecode\_disasm.h, [67](#)

OP\_LGDT  
    bytecode\_disasm.h, [67](#)

OP\_LGS  
    bytecode\_disasm.h, [67](#)

OP\_LIDT  
    bytecode\_disasm.h, [67](#)

OP\_LLDT  
    bytecode\_disasm.h, [67](#)

OP\_LODSB  
    bytecode\_disasm.h, [67](#)

OP\_LODSD  
    bytecode\_disasm.h, [67](#)

OP\_LODSW  
    bytecode\_disasm.h, [67](#)

OP\_LOOP  
    bytecode\_disasm.h, [67](#)

OP\_LOOPE  
    bytecode\_disasm.h, [68](#)

OP\_LOOPNE  
    bytecode\_disasm.h, [68](#)

OP\_LSL  
    bytecode\_disasm.h, [68](#)

OP\_LSS  
    bytecode\_disasm.h, [68](#)

OP\_LTR  
    bytecode\_disasm.h, [68](#)

OP\_MOV  
    bytecode\_disasm.h, [68](#)

OP\_MOVSB  
    bytecode\_disasm.h, [68](#)

OP\_MOVSD  
    bytecode\_disasm.h, [68](#)

OP\_MOVSW  
    bytecode\_disasm.h, [68](#)

OP\_MOVSX  
    bytecode\_disasm.h, [68](#)

OP\_MOVZX  
    bytecode\_disasm.h, [68](#)

OP\_MUL  
    bytecode\_disasm.h, [68](#)

OP\_NEG  
    bytecode\_disasm.h, [68](#)

OP\_NOP  
    bytecode\_disasm.h, [68](#)

OP\_NOT  
    bytecode\_disasm.h, [68](#)

OP\_OR  
    bytecode\_disasm.h, [68](#)

OP\_OUT  
    bytecode\_disasm.h, [68](#)

OP\_OUTSB  
    bytecode\_disasm.h, [68](#)

OP\_OUTSD  
    bytecode\_disasm.h, [68](#)

OP\_OUTSW  
    bytecode\_disasm.h, [68](#)

OP\_POP  
    bytecode\_disasm.h, [68](#)

OP\_POPAD  
    bytecode\_disasm.h, [68](#)

OP\_POPFD  
    bytecode\_disasm.h, [68](#)

OP\_PREFIX\_LOCK  
    bytecode\_disasm.h, [67](#)

OP\_PREFIX\_REPE  
    bytecode\_disasm.h, [68](#)

OP\_PREFIX\_REPNE  
    bytecode\_disasm.h, [68](#)

OP\_PUSH  
    bytecode\_disasm.h, [68](#)

OP\_PUSHAD  
    bytecode\_disasm.h, [68](#)

OP\_PUSHFD  
    bytecode\_disasm.h, [68](#)

OP\_RCL  
    bytecode\_disasm.h, [68](#)

OP\_RCR  
    bytecode\_disasm.h, [68](#)

OP\_RDMSR  
    bytecode\_disasm.h, [68](#)

OP\_RDPMC  
    bytecode\_disasm.h, [68](#)

OP\_RDTSC  
    bytecode\_disasm.h, [68](#)

OP\_RETF  
    bytecode\_disasm.h, [68](#)

OP\_RETN  
    bytecode\_disasm.h, [68](#)

OP\_ROL  
    bytecode\_disasm.h, [68](#)

OP\_ROR  
    bytecode\_disasm.h, [68](#)

OP\_RSM  
    bytecode\_disasm.h, [68](#)

OP\_SAHF  
    bytecode\_disasm.h, [68](#)

OP\_SAR  
    bytecode\_disasm.h, [68](#)

OP\_SBB  
    bytecode\_disasm.h, [68](#)

OP\_SCASB  
    bytecode\_disasm.h, [68](#)

OP\_SCASD  
    bytecode\_disasm.h, [68](#)

OP\_SCASW  
    bytecode\_disasm.h, [68](#)

- OP\_SETA
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETBE
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETC
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETG
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETGE
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETL
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETLE
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETNC
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETNO
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETNP
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETNS
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETNZ
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETO
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETP
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETS
  - [bytecode\\_disasm.h, 69](#)
- OP\_SETZ
  - [bytecode\\_disasm.h, 69](#)
- OP\_SGDT
  - [bytecode\\_disasm.h, 69](#)
- OP\_SHL
  - [bytecode\\_disasm.h, 69](#)
- OP\_SHLD
  - [bytecode\\_disasm.h, 69](#)
- OP\_SHR
  - [bytecode\\_disasm.h, 69](#)
- OP\_SHRD
  - [bytecode\\_disasm.h, 69](#)
- OP\_SIDT
  - [bytecode\\_disasm.h, 69](#)
- OP\_SLDT
  - [bytecode\\_disasm.h, 69](#)
- OP\_STC
  - [bytecode\\_disasm.h, 69](#)
- OP\_STD
  - [bytecode\\_disasm.h, 69](#)
- OP\_STI
  - [bytecode\\_disasm.h, 69](#)
- OP\_STOSB
  - [bytecode\\_disasm.h, 69](#)
- OP\_STOSD
  - [bytecode\\_disasm.h, 69](#)
- OP\_STOSW
  - [bytecode\\_disasm.h, 69](#)
- OP\_STR
  - [bytecode\\_disasm.h, 69](#)
- OP\_SUB
  - [bytecode\\_disasm.h, 69](#)
- OP\_SYSCALL
  - [bytecode\\_disasm.h, 69](#)
- OP\_SYSENTER
  - [bytecode\\_disasm.h, 69](#)
- OP\_SYSEXIT
  - [bytecode\\_disasm.h, 69](#)
- OP\_SYSRET
  - [bytecode\\_disasm.h, 69](#)
- OP\_TEST
  - [bytecode\\_disasm.h, 69](#)
- OP\_UD2
  - [bytecode\\_disasm.h, 69](#)
- OP\_VERR
  - [bytecode\\_disasm.h, 69](#)
- OP\_VERRW
  - [bytecode\\_disasm.h, 69](#)
- OP\_WBINVD
  - [bytecode\\_disasm.h, 69](#)
- OP\_WRMSR
  - [bytecode\\_disasm.h, 69](#)
- OP\_XADD
  - [bytecode\\_disasm.h, 69](#)
- OP\_XCHG
  - [bytecode\\_disasm.h, 69](#)
- OP\_XLAT
  - [bytecode\\_disasm.h, 69](#)
- OP\_XOR
  - [bytecode\\_disasm.h, 69](#)
- offset
  - [cli\\_exe\\_info, 51](#)
- operation\_size
  - [DIS\\_fixed, 54](#)
- opt32
  - [cli\\_pe\\_hook\\_data, 53](#)
- opt64
  - [cli\\_pe\\_hook\\_data, 53](#)
- other
  - [DIS\\_arg, 54](#)
- overlays
  - [cli\\_pe\\_hook\\_data, 53](#)
- overlays\_sz
  - [cli\\_pe\\_hook\\_data, 53](#)
- PDF Handling
  - [PDF\\_PHASE\\_END, 32](#)
  - [PDF\\_PHASE\\_PARSED, 32](#)
  - [PDF\\_PHASE\\_POSTDUMP, 32](#)
  - [PDF\\_PHASE\\_PRE, 32](#)
- PDF\_PHASE\_END
  - [PDF Handling, 32](#)
- PDF\_PHASE\_PARSED
  - [PDF Handling, 32](#)
- PDF\_PHASE\_POSTDUMP
  - [PDF Handling, 32](#)
- PDF\_PHASE\_PRE
  - [PDF Handling, 32](#)

- PDF Handling, 32
- PE\_INVALID\_RVA
  - bytecode\_api.h, 62
- PDF Handling, 32
  - pdf\_flag, 32
  - pdf\_get\_dumpedobjid, 32
  - pdf\_get\_flags, 32
  - pdf\_get\_obj\_num, 33
  - pdf\_get\_offset, 33
  - pdf\_get\_phase, 33
  - pdf\_getobj, 33
  - pdf\_getobjflags, 33
  - pdf\_getobjid, 34
  - pdf\_getobjsize, 34
  - pdf\_lookupobj, 34
  - pdf\_objflags, 32
  - pdf\_phase, 32
  - pdf\_set\_flags, 34
  - pdf\_setobjflags, 34
- PDF\_HOOK\_DECLARE
  - Bytecode Configuration, 10
- PE Operations, 37
  - get\_pe\_section, 38
  - getEntryPoint, 38
  - getExeOffset, 38
  - getImageBase, 38
  - getNumberOfSections, 38
  - getPEBaseOfCode, 39
  - getPEBaseOfData, 39
  - getPECharacteristics, 39
  - getPEChecksum, 39
  - getPEDataDirRVA, 39
  - getPEDataDirSize, 39
  - getPEDllCharacteristics, 40
  - getPEFileAlignment, 40
  - getPEImageBase, 40
  - getPELFANew, 40
  - getPELoaderFlags, 40
  - getPEMachine, 40
  - getPEMajorImageVersion, 41
  - getPEMajorLinkerVersion, 41
  - getPEMajorOperatingSystemVersion, 41
  - getPEMajorSubsystemVersion, 41
  - getPEMinorImageVersion, 41
  - getPEMinorLinkerVersion, 41
  - getPEMinorOperatingSystemVersion, 41
  - getPEMinorSubsystemVersion, 42
  - getPENumberOfSymbols, 42
  - getPEPointerToSymbolTable, 42
  - getPESectionAlignment, 42
  - getPESizeOfCode, 42
  - getPESizeOfHeaders, 42
  - getPESizeOfHeapCommit, 42
  - getPESizeOfHeapReserve, 43
  - getPESizeOfImage, 43
  - getPESizeOfInitializedData, 43
  - getPESizeOfOptionalHeader, 43
  - getPESizeOfStackCommit, 43
  - getPESizeOfStackReserve, 43
  - getPESizeOfUninitializedData, 43
  - getPESubsystem, 43
  - getPETimeDateStamp, 44
  - getPEWin32VersionValue, 44
  - getPEisDLL, 40
  - getSectionRVA, 44
  - getSectionVirtualSize, 44
  - getVirtualEntryPoint, 44
  - hasExeInfo, 44
  - hasPEInfo, 44
  - isPE64, 45
  - pe\_rawaddr, 45
  - readPESectionName, 45
  - readRVA, 45
- PE\_HOOK\_DECLARE
  - Bytecode Configuration, 10
- PE\_UNPACKER\_DECLARE
  - Bytecode Configuration, 10
- pdf\_flag
  - PDF Handling, 32
- pdf\_get\_dumpedobjid
  - PDF Handling, 32
- pdf\_get\_flags
  - PDF Handling, 32
- pdf\_get\_obj\_num
  - PDF Handling, 33
- pdf\_get\_offset
  - PDF Handling, 33
- pdf\_get\_phase
  - PDF Handling, 33
- pdf\_getobj
  - PDF Handling, 33
- pdf\_getobjflags
  - PDF Handling, 33
- pdf\_getobjid
  - PDF Handling, 34
- pdf\_getobjsize
  - PDF Handling, 34
- pdf\_lookupobj
  - PDF Handling, 34
- pdf\_objflags
  - PDF Handling, 32
- pdf\_phase
  - PDF Handling, 32
- pdf\_set\_flags
  - PDF Handling, 34
- pdf\_setobjflags
  - PDF Handling, 34
- pe\_image\_data\_dir, 55
- pe\_image\_file\_hdr, 55
  - Machine, 56
  - Magic, 56
  - NumberOfSections, 56
  - NumberOfSymbols, 56
  - PointerToSymbolTable, 56
  - SizeOfOptionalHeader, 56
  - TimeDateStamp, 56

- pe\_image\_optional\_hdr32, 56
  - Checksum, 57
  - FileAlignment, 57
  - ImageBase, 57
  - MajorImageVersion, 57
  - MajorLinkerVersion, 57
  - MajorOperatingSystemVersion, 57
  - MinorImageVersion, 57
  - MinorLinkerVersion, 57
  - MinorOperatingSystemVersion, 57
  - NumberOfRvaAndSizes, 57
  - SectionAlignment, 57
  - SizeOfCode, 57
  - SizeOfInitializedData, 57
  - SizeOfUninitializedData, 57
- pe\_image\_optional\_hdr64, 58
  - Checksum, 58
  - FileAlignment, 58
  - ImageBase, 58
  - MajorImageVersion, 58
  - MajorLinkerVersion, 58
  - MajorOperatingSystemVersion, 58
  - MinorImageVersion, 58
  - MinorLinkerVersion, 58
  - MinorOperatingSystemVersion, 58
  - NumberOfRvaAndSizes, 59
  - SectionAlignment, 59
  - SizeOfCode, 59
  - SizeOfInitializedData, 59
  - SizeOfUninitializedData, 59
- pe\_image\_section\_hdr, 59
  - Name, 59
  - NumberOfLinenumbers, 59
  - NumberOfRelocations, 59
  - PointerToLinenumbers, 59
  - PointerToRawData, 59
  - PointerToRelocations, 60
  - SizeOfRawData, 60
- pe\_rawaddr
  - PE Operations, 45
- PointerToLinenumbers
  - pe\_image\_section\_hdr, 59
- PointerToRawData
  - pe\_image\_section\_hdr, 59
- PointerToRelocations
  - pe\_image\_section\_hdr, 60
- PointerToSymbolTable
  - pe\_image\_file\_hdr, 56
- raw
  - cli\_exe\_section, 52
- read
  - File Operations, 25
- read\_number
  - File Operations, 26
- readPESectionName
  - PE Operations, 45
- readRVA
  - PE Operations, 45
- reg
  - DIS\_arg, 54
- res\_addr
  - cli\_exe\_info, 51
- rsz
  - cli\_exe\_section, 52
- running\_on\_jit
  - Engine Queries, 18
- rva
  - cli\_exe\_section, 52
- SEEK\_CUR
  - File Operations, 23
- SEEK\_END
  - File Operations, 23
- SEEK\_SET
  - File Operations, 23
- SIZEB
  - bytecode\_disasm.h, 65
- SIZED
  - bytecode\_disasm.h, 65
- SIZEF
  - bytecode\_disasm.h, 65
- SIZEPTR
  - bytecode\_disasm.h, 65
- SIZEQ
  - bytecode\_disasm.h, 65
- SIZET
  - bytecode\_disasm.h, 65
- SIZEW
  - bytecode\_disasm.h, 65
- SIGNATURES\_DECL\_END
  - Bytecode Configuration, 10
- SIGNATURES\_DEF\_END
  - Bytecode Configuration, 10
- SIGNATURES\_END
  - bytecode\_local.h, 74
- scale
  - DIS\_mem\_arg, 55
- scale\_reg
  - DIS\_mem\_arg, 55
- Scan Control, 46
  - bytecode\_rt\_error, 46
  - extract\_new, 46
  - extract\_set\_container, 46
  - foundVirus, 46
  - input\_switch, 47
  - setvirusname, 47
- section
  - cli\_exe\_info, 51
- SectionAlignment
  - pe\_image\_optional\_hdr32, 57
  - pe\_image\_optional\_hdr64, 59
- seek
  - File Operations, 26
- segment
  - DIS\_fixed, 54
- setvirusname
  - Scan Control, 47

- SizeOfCode
  - pe\_image\_optional\_hdr32, [57](#)
  - pe\_image\_optional\_hdr64, [59](#)
- SizeOfInitializedData
  - pe\_image\_optional\_hdr32, [57](#)
  - pe\_image\_optional\_hdr64, [59](#)
- SizeOfOptionalHeader
  - pe\_image\_file\_hdr, [56](#)
- SizeOfRawData
  - pe\_image\_section\_hdr, [60](#)
- SizeOfUninitializedData
  - pe\_image\_optional\_hdr32, [57](#)
  - pe\_image\_optional\_hdr64, [59](#)
- String Operations, [48](#)
  - atoi, [48](#)
  - entropy\_buffer, [48](#)
  - hex2ui, [48](#)
  - memchr, [49](#)
  - memcmp, [49](#)
  - memcpy, [49](#)
  - memmove, [49](#)
  - memset, [50](#)
  - memstr, [50](#)
- TARGET
  - Bytecode Configuration, [11](#)
- test1
  - bytecode\_api.h, [62](#)
- test2
  - bytecode\_api.h, [62](#)
- TimeStamp
  - pe\_image\_file\_hdr, [56](#)
- uraw
  - cli\_exe\_section, [52](#)
- ursz
  - cli\_exe\_section, [52](#)
- urva
  - cli\_exe\_section, [52](#)
- uvsz
  - cli\_exe\_section, [52](#)
- VIRUSNAME\_PREFIX
  - Bytecode Configuration, [11](#)
- VIRUSNAMES
  - Bytecode Configuration, [11](#)
- version\_compare
  - Environment, [22](#)
- vsz
  - cli\_exe\_section, [52](#)
- write
  - File Operations, [26](#)
- x86\_opcode
  - DIS\_fixed, [54](#)
- X86OPS
  - bytecode\_disasm.h, [65](#)
- X86REGS
  - bytecode\_disasm.h, [71](#)