YABE

version 0 release 0

Generated by Doxygen 1.7.6.1

Tue Oct 2 2012 10:55:51

Contents

Chapter 1

Low level C calls to write and read YABE encoded data

1.1 Introduction

YABE is the acronym of Yet Another Binary Encoding. The type of values it may encode is limited to the one of Javascript, including the *blob* type. It is a superset of the JSON supported data type set by the addition of the *blob* value type that JSON, which is text only, can't easily and efficiently represent.

The rationale to choose the Javascript limited data set is because it is sufficient for most applications and because it makes also the encoding and decoding source code very small and easy to write, understand and check. This is also most likely the reason of the JSON encoding success.

The benefit of a binary encoding is that marshalling is faster than with JSON, because *blob* values are naturally represented in it and binary encoding is slightly more compact than text only encoding.

This encoding has been named YABE because there already exist a few encodings with similar proporties around. YABE distinguishes itself from them by its encoding and its API.

Remarks

This code taged YABE_v0_r0 is version 0 release 0 of the C source code.

1.2 Data type set

The data set is the one defined for Javascript. It is the data set supported by JSON which uses a text only encoding, with the addition of the *blob* value type that is also part of Javascript.

A *blob* is an array of raw bytes (binary) with a mime type string. Since YABE is binary

encoded, it can easily and efficiently encode *blob* values.

1.2.1 Atomic data values

- **null** ;
- **Boolean** : *true* & *false* ;
- **Integer** : 64 bit integer ;
- **Floating point** : 64 bit IEEE 754-2008 ;
- **String** : utf8 encoded character sequence ;
- **Blob** : array of raw bytes with a mime type string ;

1.2.2 Composed values

- **Array** : Sequence of any values ;
- **Object** : Sequence of a pair of value identifier (string) and any value ;

1.2.3 Data encoding

Each value is encoded as a tag byte identifying its type, followed by an optional value size and the value itself. When possible the size or the value are stored in the tag.

```
[tag]([size])([value])

| Value  |    Tag    | arguments        | comment
-------------------------------------------------------------------------
  0..127 : [0xxxxxxx]                   : integer value 0..127
  str6   : [10xxxxxx] [byte]*           : utf8 char string
  null   : [11000000]                   : null value
  int16  : [11000001] [int16]           : 16 bit integer
  int32  : [11000010] [int32]           : 32 bit integer
  int64  : [11000011] [int64]           : 64 bit integer
  flt0   : [11000100]                   : 0. float value
  flt16  : [11000101] [flt16]           : 16 bit float
  flt32  : [11000110] [flt32]           : 32 bit float
  flt64  : [11000111] [flt64]           : 64 bit float
  false  : [11001000]                   : boolean false value
  true   : [11001001]                   : boolean true value
  blob   : [11001010] [string] [string] : mime typed byte array
  ends   : [11001011]                   : equivalent to ] or }
  none   : [11001100]                   : tag byte to be ignored
  str16  : [11001101] [len16] [byte]*   : utf8 char string
  str32  : [11001110] [len32] [byte]*   : utf8 char string
  str64  : [11001111] [len64] [byte]*   : utf8 char string
  sarray : [11010xxx] [value]*          : 0 to 6 value array
  arrays : [11010111] [value]*          : equivalent to [
  sobject: [11011xxx] [str,value]*      : 0 to 6 value object
  objects: [11011111] [str,value]*      : equivalent to {
  -1..-32: [111xxxxx]                   : integer value -1..-32
```

- The tag is a one byte value ;
- Integer values from -32 to 127 are encoded as is as the tag value ;
- Integer values are encoded as little endian signed integer ;
- Floating point values are encoded in the IEEE 754-2008 format (half, float, double) ;
- A strings is a sequence of utf8 encoded chars with the number of bytes as length ;
- A length value is encoded as little endian unsigned integer of 16, 32 or 64 bits ;
- A blob is a pair of strings, the first is a mime type and the second is a sequence of raw bytes ;
- An Array is encoded as a stream of values ;
- An Object is encoded as a stream of string and value pairs where the string is a unique identifier ;
- An Object may not have an empty string as identifier ;
- An array or an object stream is ended by the *ends* tag ;
- If an array or an object have less than 7 items, the *sarray* or *sobject* encoding should be used where the number of items is encoded in the tag and there is no *ends* tag ;

1.2.4 YABE encoded block signature

A 5 byte signature may start a byte block containing YABE encoded data. The first four bytes are the ASCII code 'Y', 'A', 'B', 'E' in that order, and the fifth byte is the version number of the encoding. This short specification describes the encoding version 0.

Remarks

The size of a YABE encoded data block must be determined by the context.

1.3 YABE writing and reading API

The yabe.h and yabe.c files provide low level C functions to write and read YABE encoded data. The provided code doesn't manage the buffer storage because there are too many different ways to do this.

The user may want to grow the buffer as needed, append a new buffer block to a chain of block, send through the network or write to file the filled buffer and resume with the buffer emptied, etc.

In the C API functions, YABE writing and reading use a *cursor* to keep track where to write or read data in memory. When writing, the cursor holds a pointer on position

in memory where to write and the number of free writable bytes in the buffer. When reading, the cursor holds a pointer on the position in memory where the next data to read is located and the number of bytes left to read in the buffer. The user is responsible to initialize the cursor accordingly.

All reading and writing functions return the number of bytes read or written. The operation has thus failed if the returned value is 0. When reading, the end of buffer should have been reached. If not, then an error occured in the decoding.

1.3.1 YABE data writing

All YABE encoded values are written in contiguous bytes in the buffer. This is called atomic values writing. If there is not enough room in the buffer to write the value bytes, the operation failes and return 0 as the number of bytes written. Atomic values can be at most 9 bytes long.

The only exception is the yabe_write_data() functions which writes as much data bytes as possible and will thus never fail. If all the bytes could not be written, the user must resume the operation by a new call to write the remaining data bytes when new buffer space has been made available.

If YABE encoded data is to be written in an array of buffers with predefined fixed size, the remaining space of the buffer must be padded with *none* value which needs only a single byte as storage space. *None* values are silently skipped when reading YABE encoded data.

1.3.2 YABE data reading

Since any type of value can be stored at any position in a YABE encoded stream, this API should be used in the following way

1. while there is data left to read from the YABE encoded stream ;
2. for each possible type of data with *none* value as a last resort ;
3. try reading the value
4. if the return value is not 0 (it succeeded), resume with step 1 to read the next value ;
5. otherwise a fatal error occured for one of the following reasons which can't be distinguished:
    - the value to read has been trucated ;
    - a previous decoding error made reading out of sync ;
    - data is not YABE encoded data.

Some values implies to be followed by a number of other values, sometimes with a well defined type. It is the case for blobs that must be followed by two strings and for arrays and objects as well. It is the user responsibility to very that this implicit rules are respected.

1.4 Examples

1.4.1 Writing some values.

```
// Some buffer with enough space
const size_t bufLen = 1024;
char * buffer[bufLen];

// A cursor where to write into the buffer
yabe_cursor_t wCur = { buffer, bufLen };
// msgLen keeps tracks of encoded data byte length, res if to check results
size_t msgLen = 0, res;

// Write yabe encoded data signature (a 5 byte constant with version)
msgLen += res = yabe_write_signature( &wCur );
if( !res ) { ... not done because buffer would overflow ... }

// Write a null value (will be coded into one byte)
msgLen += res = yabe_write_null( &wCur );
if( !res ) { ... not done because buffer would overflow ... }

// Write a small integer value (will be coded into one byte)
msgLen += res = yabe_write_integer( &wCur, 123 );
if( !res ) { ... not done because buffer would overflow ... }

// Write a floating point value (will be coded into three byte)
msgLen += res = yabe_write_float( &wCur, 8.5 );
if( !res ) { ... not done because buffer would overflow ... }

// Write a string value
char* aString = "test string";
size_t strLen = strlen( aString ) + 1; // include trailing '\0'
msgLen += res = yabe_write_string( &wCur, strLen );
if( !res ) { ... not done because buffer would overflow ... }
msgLen += res = yabe_write_data( &wCur, aString, strLen );
if( res != strLen ) { ... string only partially written ... }

// msgLen is the number of bytes of encoded data which starts
// at buffer[0];
```

In case a writing fails, it means the data doesn't fit in the remaing free space of the buffer. The user could then grow the buffer, chain another buffer, or send or write the data to file the buffer and reset wCur to the start of buffer.

If the data is encoded in a sequence of fixed size buffers referenced by an iovec structure for instance, the unused remaining space of buffers must be padded with *none* values so that these bytes will be skipped when reading the encoded data. The following code example shows how to do that.

```
// Padding a buffer with none values
while( !yabe_end_of_buffer( &wCur ) )
    yabe_write_none( &wCur );
```

1.4.2 Reading some values

The following example illustrates how to read different types of values. However when reading YABE encoded data, the user should implement YABE data reading "this algo-

rithm".

```
// set cursor where to read data
yabe_cursor_t rCur = { data, size };
size_t res;

// Check yabe encoded data signature (a 5 byte constant with version)
res = yabe_read_signature( &rCur );
if( res == 0 ) { ... invalid yabe signature or end of buffer reached ... }
else if( res == 4 ) { ... invalid yabe encoding version ... }
assert( res == 5 );

// Read a null value (will be coded into one byte)
res = yabe_read_null( &rCur );
if( !res ) { ... next value is not null or end of buffer reached ... }
assert( res == 1 );

// Read an integer value
int64_t intValue;
res = yabe_write_integer( &rCur, &intValue );
if( !res ) { ...  next value is not integer or end of buffer reached  ... }
assert( res == 1 || res == 3 || res == 5 || res == 9 );

// Read a floating point value
res = yabe_write_float( &rCur, 8.5 );
if( !res ) { ... next value is not a float or end of buffer reached ... }
assert( res == 1 || res == 3 || res == 5 || res == 9 );

// Read a string
size_t strLen;
res = yabe_read_string( &rCur, &strLen ); // read the string length
if( !res ) { ... next value is not a string or end of buffer reached ... }
assert( res == 1 || res == 3 || res == 5 || res == 9 );
char* aString = malloc( strLen ); // get a storage for the string
res = yabe_read_data( &rCur, aString, strLen );
if( res != strLen ) { ... string partially read, end of buffer reached ...
  }

// Check if end of buffer reached
if( yabe_end_of_buffer( &rCur ) ) { ... end of buffer is reached ... }
else { ... there is some more data ... }
```

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

File Documentation

3.1 yabe.h File Reference

`#include <stdint.h> #include <ctype.h> #include <string.-h> #include <assert.h> #include <stdbool.h>`

Data Structures

- struct yabe_cursor_t

  Cursor in buffer where yabe encoded data is to be written or read. More...

Functions

- static size_t yabe_write_none (yabe_cursor_t *cursor)

  Tries writing a none value and returns the number of bytes written.
- static size_t yabe_write_null (yabe_cursor_t *cursor)

  Tries writing a null value and returns the number of bytes written.
- static size_t yabe_write_bool (yabe_cursor_t *cursor, bool value)

  Tries writing the boolean value and returns the number of bytes written.
- size_t yabe_write_integer (yabe_cursor_t *cursor, int64_t value)

  Tries writing the integer value and returns the number of bytes written.
- size_t yabe_write_float (yabe_cursor_t *cursor, double value)

  Tries writing the double float value and returns the number of bytes written.
- size_t yabe_write_string (yabe_cursor_t *cursor, size_t byteSize)

  Tries writing the string tag and its byte size values and returns the number of bytes written.
- static size_t yabe_write_data (yabe_cursor_t *cursor, const void *data, size_t size)

  Writes as much bytes as possible at cursor position until all bytes of the sequence are written or the end of buffer is met, and returns the number of bytes written.

- static size_t yabe_write_blob (yabe_cursor_t *cursor)

  Tries writing a blob tag and returns the number of bytes written.
- static size_t yabe_write_small_array (yabe_cursor_t *cursor, size_t nbr)

  Tries writing a small array tag and returns the number of bytes written.
- static size_t yabe_write_array_stream (yabe_cursor_t *cursor)

  Tries writing an array stream tag and returns the number of bytes written.
- static size_t yabe_write_small_object (yabe_cursor_t *cursor, size_t nbr)

  Tries writing a small object tag and returns the number of bytes written.
- static size_t yabe_write_object_stream (yabe_cursor_t *cursor)

  Tries writing an object stream tag and returns the number of bytes written.
- static size_t yabe_write_end_stream (yabe_cursor_t *cursor)

  Tries writing the end stream tag and returns the number of bytes written.
- static size_t yabe_write_signature (yabe_cursor_t *cursor)

  Tries writing the yabe signature ['Y','A','B','E', 0] and returns the number of bytes written.
- static bool yabe_end_of_buffer (const yabe_cursor_t *cursor)

  Return true if the end of buffer is reached, false otherwise.
- static size_t yabe_read_none (yabe_cursor_t *cursor)

  Skip none value in buffer if any and returns the number of bytes skipped.
- static size_t yabe_read_null (yabe_cursor_t *cursor)

  Try reading the value as null and returns the number of byte read.
- static size_t yabe_read_bool (yabe_cursor_t *cursor, bool *value)

  Try reading the value as a boolean and returns the number of byte read.
- size_t yabe_read_integer (yabe_cursor_t *cursor, int64_t *value)

  Try reading the value as an integer, skipping subsequent none values if any, and returns the number of byte read.
- size_t yabe_read_float (yabe_cursor_t *cursor, double *value)

  Try reading the value as a double float and returns the number of byte read.
- size_t yabe_read_string (yabe_cursor_t *cursor, size_t *length)

  Try reading the value as a string and returns the number of byte read.
- static size_t yabe_read_data (yabe_cursor_t *cursor, void *data, size_t size)

  Try reading the requested number of data bytes at the cursor position and returns the number of bytes effectively read.
- static size_t yabe_read_blob (yabe_cursor_t *cursor)

  Try reading the value as blob and returns the number of byte read.
- static size_t yabe_read_small_array (yabe_cursor_t *cursor, int8_t *number)

  Try reading the value as a small array and returns the number of byte read.
- static size_t yabe_read_small_object (yabe_cursor_t *cursor, int8_t *number)

  Try reading the value as a small object and returns the number of byte read.
- static size_t yabe_read_array_stream (yabe_cursor_t *cursor)

  Try reading the value as an array stream, skipping subsequent none values if any, and returns the number of byte read.
- static size_t yabe_read_object_stream (yabe_cursor_t *cursor)

Try reading the value as an object stream, skipping subsequent none values if any, and returns the number of byte read.

- static size_t yabe_read_end_stream (yabe_cursor_t *cursor)

  Try reading the value as an end of stream, skipping subsequent none values if any, and returns the number of byte read.

- static size_t yabe_read_signature (yabe_cursor_t *cursor)

  Try reading the yabe signature ['Y','A','B','E', 0].

3.1.1 Data Structure Documentation

3.1.1.1 struct yabe_cursor_t

Cursor in buffer where yabe encoded data is to be written or read.

Definition at line 305 of file yabe.h.

Data Fields

char *	ptr	Pointer on next byte to read or where to write.
size_t	len	Number of bytes left to read or to write.

3.1.2 Function Documentation

3.1.2.1 static size_t yabe_write_none (yabe_cursor_t * *cursor*) [inline, static]

Tries writing a *none* value and returns the number of bytes written.

The purpose of this value is to be used as padding or to overwrite encoded values so that they become ignored when reading.

Note

This value encoded in one byte is silently skipped when reading.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if value could be written

Returns

the number of bytes written, *fail* : 0, *success* : 1

Definition at line 377 of file yabe.h.

3.1.2.2 static size_t yabe_write_null (yabe_cursor_t * *cursor*) [inline, static]

Tries writing a *null* value and returns the number of bytes written.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if the value could be written

Returns

the number of bytes written, *fail* : 0, *success* : 1

Definition at line 388 of file yabe.h.

3.1.2.3 static size_t yabe_write_bool (yabe_cursor_t * *cursor,* bool *value*) [inline, static]

Tries writing the boolean value and returns the number of bytes written.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if value could be written
	value	Boolean value to try writing at cursor position

Returns

the number of bytes written, *fail* : 0, *success* : 1

Definition at line 400 of file yabe.h.

3.1.2.4 size_t yabe_write_integer (yabe_cursor_t * *cursor,* int64_t *value*)

Tries writing the integer value and returns the number of bytes written.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if value could be written
	value	64bit integer value to try writing at cursor position

Returns

the number of bytes written, *fail* : 0, *success* : 1, 3, 5 or 9

3.1.2.5 size_t yabe_write_float (yabe_cursor_t * *cursor,* double *value*)

Tries writing the double float value and returns the number of bytes written.

Denormalized floating point values are rounded to 0.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if value could be written
	value	64bit floating point value to try writing at cursor position

Returns

the number of bytes written, *fail* : 0, *success* : 1, 3, 5 or 9

3.1.2.6 size_t yabe_write_string (yabe_cursor_t * *cursor,* size_t *byteSize*)

Tries writing the string tag and its byte size values and returns the number of bytes written.

Only the string value tag and the string byte size are written at the cursor position. The string bytes (utf8 chars) must be written using the yabe_write_data() function.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if value could be written
	byteSize	byte length of the utf8 encoded string to try writing at cursor position

Returns

the number of bytes written, *fail* : 0, *success* : 1, 3, 5 or 9

3.1.2.7 static size_t yabe_write_data (yabe_cursor_t * *cursor,* const void * *data,* size_t *size*) [inline, static]

Writes as much bytes as possible at cursor position until all bytes of the sequence are written or the end of buffer is met, and returns the number of bytes written.

The data may be partially written. A returned value smaller than *size* means the data could not be fully written. One or more additionnal calls to this function are required to write the remaining bytes of data.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if value could be written

	data	Pointer on the bytes to writing at cursor position
	size	Number of bytes to write at cursor position

Returns

the number of bytes written, *incomplete* : $<$ *size*, *complete* : *size*

Definition at line 463 of file yabe.h.

References yabe_cursor_t::len, and yabe_cursor_t::ptr.

Referenced by yabe_write_signature().

3.1.2.8 static size_t yabe_write_blob (yabe_cursor_t * *cursor*) [inline, static]

Tries writing a blob tag and returns the number of bytes written.

A blob *must* be followed by two strings. The first string encodes the mime type of the blob data. The second string contains the blob data made of raw bytes. The second string doesn't necessarily contain utf8 chars.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if the value could be written

Returns

the number of bytes written, *fail* : 0, *success* : 1

Definition at line 487 of file yabe.h.

3.1.2.9 static size_t yabe_write_small_array (yabe_cursor_t * *cursor,* size_t *nbr*) [inline, static]

Tries writing a small array tag and returns the number of bytes written.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if the value could be written
	nbr	Number of values in array : 0$<$= nbr $<$= 6

Returns

the number of bytes written, *fail* : 0, *success* : 1

Definition at line 499 of file yabe.h.

3.1.2.10 static size_t yabe_write_array_stream (yabe_cursor_t * *cursor*) [inline, static]

Tries writing an array stream tag and returns the number of bytes written.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if the value could be written

Returns

the number of bytes written, *fail* : 0, *success* : 1

Definition at line 510 of file yabe.h.

3.1.2.11 static size_t yabe_write_small_object (yabe_cursor_t * *cursor,* size_t *nbr*) [inline, static]

Tries writing a small object tag and returns the number of bytes written.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if the value could be written
	nbr	Number of identfier, values pairs in object : 0<= nbr <= 6

Returns

the number of bytes written, *fail* : 0, *success* : 1

Definition at line 522 of file yabe.h.

3.1.2.12 static size_t yabe_write_object_stream (yabe_cursor_t * *cursor*) [inline, static]

Tries writing an object stream tag and returns the number of bytes written.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if the value could be written

Returns

the number of bytes written, *fail* : 0, *success* : 1

Definition at line 533 of file yabe.h.

3.1.2.13 static size_t yabe_write_end_stream (yabe_cursor_t * *cursor*) [inline, static]

Tries writing the end stream tag and returns the number of bytes written.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if the value could be written

Returns

the number of bytes written, *fail* : 0, *success* : 1

Definition at line 544 of file yabe.h.

3.1.2.14 static size_t yabe_write_signature (yabe_cursor_t * *cursor*) [inline, static]

Tries writing the yabe signature ['Y','A','B','E', 0] and returns the number of bytes written.

Parameters

in,out	*cursor*	Pointer on buffer info where to write value, update it if the value could be written

Returns

the number of bytes written, *fail* : 0, *success* : 5

Definition at line 556 of file yabe.h.

References yabe_cursor_t::len, and yabe_write_data().

3.1.2.15 static bool yabe_end_of_buffer (const yabe_cursor_t * *cursor*) [inline, static]

Return true if the end of buffer is reached, false otherwise.

Parameters

cursor	Pointer on buffer to test

Returns

true if the end of buffer is reached, false otherwise

Definition at line 574 of file yabe.h.

References yabe_cursor_t::len.

3.1.2.16 static size_t yabe_read_none (yabe_cursor_t * *cursor*) [inline, static]

Skip *none* value in buffer if any and returns the number of bytes skipped.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated only if *none* tags where skipped

Returns

the number of bytes skipped

Definition at line 636 of file yabe.h.

References yabe_cursor_t::len, and yabe_cursor_t::ptr.

3.1.2.17 static size_t yabe_read_null (yabe_cursor_t * *cursor*) [inline, static]

Try reading the value as null and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds

Returns

the number of bytes read, *fail* : 0, *success* : 1

Definition at line 659 of file yabe.h.

3.1.2.18 static size_t yabe_read_bool (yabe_cursor_t * *cursor*, bool * *value*) [inline, static]

Try reading the value as a boolean and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds
out	*value*	the boolean value if the value is a boolean, otherwise the value is left unchanged

Returns

the number of bytes read, *fail* : 0, *success* : 1

Definition at line 674 of file yabe.h.

3.1.2.19 size_t yabe_read_integer (yabe_cursor_t * *cursor*, int64_t * *value*)

Try reading the value as an integer, skipping subsequent *none* values if any, and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds
out	*value*	the integer value if the value is an integer, otherwise the value is left unchanged

Returns

the number of bytes read, *fail* : 0, *success* : 1, 3, 5, 9

3.1.2.20 size_t yabe_read_float (yabe_cursor_t * *cursor*, double * *value*)

Try reading the value as a double float and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called. Assume double or float is the IEEE 754 double or float representation

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds
out	*value*	the double float value if the value is a double float, otherwise the value is left unchanged

Returns

the number of bytes read, *fail* : 0, *success* : 1, 3, 5, 9

3.1.2.21 size_t yabe_read_string (yabe_cursor_t * *cursor*, size_t * *length*)

Try reading the value as a string and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called.

Remarks

This function only reads the string byte length if it succeeds. A subsequent call to yabe_read_data() is required to read the string bytes.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds
out	*length*	the string byte length if the value is a string, otherwise the length value is left unchanged

Returns

the number of bytes read, *fail* : 0, *success* : 1, 3, 5, 9

3.1.2.22 static size_t yabe_read_data (yabe_cursor_t * *cursor*, void * *data*, size_t *size*) [inline, static]

Try reading the requested number of data bytes at the cursor position and returns the number of bytes effectively read.

Once the function has read all the data bytes it was requested to read it skips all subsequent *none* value to be ready for reading the next value.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds
out	*data*	The pointer where to store data bytes read
	size	The number of bytes to read

Returns

the number of bytes read, *incomplete* : $<$ size, *complete* : size

Definition at line 748 of file yabe.h.

References yabe_cursor_t::len, and yabe_cursor_t::ptr.

3.1.2.23 static size_t yabe_read_blob (yabe_cursor_t * *cursor*) [inline, static]

Try reading the value as blob and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called.

A blob value is followed by two strings. The first string specifies the mime type of the blob data and the second string contains the raw bytes of the blob.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds

Returns

the number of bytes read, *fail* : 0, *success* : 1

Definition at line 774 of file yabe.h.

3.1.2.24 static size_t yabe_read_small_array (yabe_cursor_t * *cursor*, int8_t * *number*) [inline, static]

Try reading the value as a small array and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called.

If the returned value is none zero, it is followed by *number* YABE encoded values corresponding to the items of the array.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds
out	*number*	the number of items in the small array if the read succeeds, otherwise number is left unchanged

Returns

the number of bytes read, *fail* : 0, *success* : 1

Definition at line 792 of file yabe.h.

3.1.2.25 static size_t yabe_read_small_object (yabe_cursor_t * *cursor*, int8_t * *number*) [inline, static]

Try reading the value as a small object and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called.

If the returned value is none zero, it is followed by *number* of string and YABE encoded values pairs corresponding to the items of the object.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds
out	*number*	the number of items in the small object if the read succeeds, otherwise number is left unchanged

Returns

the number of bytes read, *fail* : 0, *success* : 1

Definition at line 817 of file yabe.h.

3.1.2.26 static size_t yabe_read_array_stream (yabe_cursor_t * *cursor*) [inline, static]

Try reading the value as an array stream, skipping subsequent *none* values if any, and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called. If the returned value is none zero, this value is followed by a sequence of values contained in the array. The sequence ends when an end stream value could be read.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds

Returns

the number of bytes read, *fail* : 0, *success* : 1

Definition at line 840 of file yabe.h.

3.1.2.27 static size_t yabe_read_object_stream (yabe_cursor_t * *cursor*) [inline, static]

Try reading the value as an object stream, skipping subsequent *none* values if any, and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called. If the returned value is none zero, this value is followed by a sequence of pairs of string,value contained in the object, where the string is the value identifier. The sequence ends when an end stream value could be read.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds

Returns

the number of bytes read, *fail* : 0, *success* : 1

Definition at line 857 of file yabe.h.

3.1.2.28 static size_t yabe_read_end_stream (yabe_cursor_t * *cursor*) [inline, static]

Try reading the value as an end of stream, skipping subsequent *none* values if any, and returns the number of byte read.

Requires the cursor is not at the end of buffer when the function is called. If the returned value is none zero, the end of array or object value stream has been reached.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds

Returns

the number of bytes read, *fail* : 0, *success* : 1

Definition at line 873 of file yabe.h.

3.1.2.29 static size_t yabe_read_signature (yabe_cursor_t * *cursor*) [inline, static]

Try reading the yabe signature ['Y','A','B','E', 0].

It requires there are at least 5 bytes to read in the buffer. Reads the first 4 bytes if they match, read also the version if it matches.

Parameters

in,out	*cursor*	Pointer on buffer where to try reading, the cursor is updated if the read operation succeeds

Returns

the number of bytes read, *fail* : 0, *bad* version : 4, *success* : 5

Definition at line 887 of file yabe.h.

References yabe_cursor_t::len, and yabe_cursor_t::ptr.