Rocky Enterprise Linux 9.2 Manual Pages on command 'OSSL_PARAM_modified.3ossl'

$ man OSSL_PARAM_modified.3ossl

OSSL_PARAM_INT(3ossl) OpenSSL OSSL_PARAM_INT(3ossl)

NAME

OSSL_PARAM_double, OSSL_PARAM_int, OSSL_PARAM_int32, OSSL_PARAM_int64,

OSSL_PARAM_long, OSSL_PARAM_size_t, OSSL_PARAM_time_t, OSSL_PARAM_uint,

OSSL_PARAM_uint32, OSSL_PARAM_uint64, OSSL_PARAM_ulong, OSSL_PARAM_BN,

OSSL_PARAM_utf8_string, OSSL_PARAM_octet_string, OSSL_PARAM_utf8_ptr,

OSSL_PARAM_octet_ptr, OSSL_PARAM_END, OSSL_PARAM_DEFN,

OSSL_PARAM_construct_double, OSSL_PARAM_construct_int,

OSSL_PARAM_construct_int32, OSSL_PARAM_construct_int64,

OSSL_PARAM_construct_long, OSSL_PARAM_construct_size_t,

OSSL_PARAM_construct_time_t, OSSL_PARAM_construct_uint,

OSSL_PARAM_construct_uint32, OSSL_PARAM_construct_uint64,

OSSL_PARAM_construct_ulong, OSSL_PARAM_construct_BN,

OSSL_PARAM_construct_utf8_string, OSSL_PARAM_construct_utf8_ptr,

OSSL_PARAM_construct_octet_string, OSSL_PARAM_construct_octet_ptr,

OSSL_PARAM_construct_end, OSSL_PARAM_locate, OSSL_PARAM_locate_const,

OSSL_PARAM_get_double, OSSL_PARAM_get_int, OSSL_PARAM_get_int32,

OSSL_PARAM_get_int64, OSSL_PARAM_get_long, OSSL_PARAM_get_size_t,

OSSL_PARAM_get_time_t, OSSL_PARAM_get_uint, OSSL_PARAM_get_uint32,

OSSL_PARAM_get_uint64, OSSL_PARAM_get_ulong, OSSL_PARAM_get_BN,

OSSL_PARAM_get_utf8_string, OSSL_PARAM_get_octet_string,

OSSL_PARAM_get_utf8_ptr, OSSL_PARAM_get_octet_ptr,

OSSL_PARAM_get_utf8_string_ptr, OSSL_PARAM_get_octet_string_ptr,

OSSL_PARAM_set_double, OSSL_PARAM_set_int, OSSL_PARAM_set_int32,

OSSL_PARAM_set_int64, OSSL_PARAM_set_long, OSSL_PARAM_set_size_t,

OSSL_PARAM_set_time_t, OSSL_PARAM_set_uint, OSSL_PARAM_set_uint32,

OSSL_PARAM_set_uint64, OSSL_PARAM_set_ulong, OSSL_PARAM_set_BN,

OSSL_PARAM_set_utf8_string, OSSL_PARAM_set_octet_string,

OSSL_PARAM_set_utf8_ptr, OSSL_PARAM_set_octet_ptr,

OSSL_PARAM_UNMODIFIED, OSSL_PARAM_modified,

OSSL_PARAM_set_all_unmodified - OSSL_PARAM helpers

SYNOPSIS

```
#include <openssl/params.h>

/*
 * TYPE in function names is one of:
 * double, int, int32, int64, long, size_t, time_t, uint, uint32, uint64, ulong
 * Corresponding TYPE in function arguments is one of:
 * double, int, int32_t, int64_t, long, size_t, time_t, unsigned int, uint32_t,
 * uint64_t, unsigned long
 */

#define OSSL_PARAM_TYPE(key, address)
#define OSSL_PARAM_BN(key, address, size)
#define OSSL_PARAM_utf8_string(key, address, size)
#define OSSL_PARAM_octet_string(key, address, size)
#define OSSL_PARAM_utf8_ptr(key, address, size)
#define OSSL_PARAM_octet_ptr(key, address, size)
#define OSSL_PARAM_END
```

```
#define OSSL_PARAM_UNMODIFIED

#define OSSL_PARAM_DEFN(key, type, addr, sz)    \
  { (key), (type), (addr), (sz), OSSL_PARAM_UNMODIFIED }

OSSL_PARAM OSSL_PARAM_construct_TYPE(const char *key, TYPE *buf);
OSSL_PARAM OSSL_PARAM_construct_BN(const char *key, unsigned char *buf,
                                   size_t bsize);
OSSL_PARAM OSSL_PARAM_construct_utf8_string(const char *key, char *buf,
                                            size_t bsize);
OSSL_PARAM OSSL_PARAM_construct_octet_string(const char *key, void *buf,
                                             size_t bsize);
OSSL_PARAM OSSL_PARAM_construct_utf8_ptr(const char *key, char **buf,
                                         size_t bsize);
OSSL_PARAM OSSL_PARAM_construct_octet_ptr(const char *key, void **buf,
                                          size_t bsize);
OSSL_PARAM OSSL_PARAM_construct_end(void);

OSSL_PARAM *OSSL_PARAM_locate(OSSL_PARAM *array, const char *key);
const OSSL_PARAM *OSSL_PARAM_locate_const(const OSSL_PARAM *array,
                                          const char *key);

int OSSL_PARAM_get_TYPE(const OSSL_PARAM *p, TYPE *val);
int OSSL_PARAM_set_TYPE(OSSL_PARAM *p, TYPE val);

int OSSL_PARAM_get_BN(const OSSL_PARAM *p, BIGNUM **val);
int OSSL_PARAM_set_BN(OSSL_PARAM *p, const BIGNUM *val);

int OSSL_PARAM_get_utf8_string(const OSSL_PARAM *p, char **val,
                               size_t max_len);
int OSSL_PARAM_set_utf8_string(OSSL_PARAM *p, const char *val);
```

```
int OSSL_PARAM_get_octet_string(const OSSL_PARAM *p, void **val,
                                size_t max_len, size_t *used_len);
int OSSL_PARAM_set_octet_string(OSSL_PARAM *p, const void *val, size_t len);

int OSSL_PARAM_get_utf8_ptr(const OSSL_PARAM *p, const char **val);
int OSSL_PARAM_set_utf8_ptr(OSSL_PARAM *p, const char *val);

int OSSL_PARAM_get_octet_ptr(const OSSL_PARAM *p, const void **val,
                             size_t *used_len);
int OSSL_PARAM_set_octet_ptr(OSSL_PARAM *p, const void *val,
                             size_t used_len);

int OSSL_PARAM_get_utf8_string_ptr(const OSSL_PARAM *p, const char **val);
int OSSL_PARAM_get_octet_string_ptr(const OSSL_PARAM *p, const void **val,
                                    size_t *used_len);

int OSSL_PARAM_modified(const OSSL_PARAM *param);
void OSSL_PARAM_set_all_unmodified(OSSL_PARAM *params);
```

DESCRIPTION

A collection of utility functions that simplify and add type safety to the OSSL_PARAM arrays. The following TYPE names are supported:

- double
- int
- int32 (int32_t)
- ?

int64 (int64_t)

?

long int (long)

?

time_t

?

size_t

?

uint32 (uint32_t)

?

uint64 (uint64_t)

?

unsigned int (uint)

?

unsigned long int (ulong)

OSSL_PARAM_TYPE() are a series of macros designed to assist initialising an array of OSSL_PARAM structures. Each of these macros defines a parameter of the specified TYPE with the provided key and parameter variable address.

OSSL_PARAM_utf8_string(), OSSL_PARAM_octet_string(), OSSL_PARAM_utf8_ptr(), OSSL_PARAM_octet_ptr(), OSSL_PARAM_BN() are macros that provide support for defining UTF8 strings, OCTET strings and big numbers. A parameter with name key is defined. The storage for this parameter is at address and is of size bytes.

OSSL_PARAM_END provides an end of parameter list marker. This should terminate all OSSL_PARAM arrays.

The OSSL_PARAM_DEFN() macro provides the ability to construct a single OSSL_PARAM (typically used in the construction of OSSL_PARAM arrays). The key, type, addr and sz arguments correspond to the key, data_type, data and data_size fields of the OSSL_PARAM structure as described on the OSSL_PARAM(3) page.

OSSL_PARAM_construct_TYPE() are a series of functions that create OSSL_PARAM records dynamically. A parameter with name key is created. The parameter will use storage pointed to by buf and return size of ret.

OSSL_PARAM_construct_BN() is a function that constructs a large integer OSSL_PARAM structure. A parameter with name key, storage buf, size bsize and return size rsize is created.

OSSL_PARAM_construct_utf8_string() is a function that constructs a UTF8 string OSSL_PARAM structure. A parameter with name key, storage buf and size bsize is created. If bsize is zero, the string length is determined using strlen(3). Generally pass zero for bsize instead of calling strlen(3) yourself.

OSSL_PARAM_construct_octet_string() is a function that constructs an OCTET string OSSL_PARAM structure. A parameter with name key, storage buf and size bsize is created.

OSSL_PARAM_construct_utf8_ptr() is a function that constructs a UTF8 string pointer OSSL_PARAM structure. A parameter with name key, storage pointer *buf and size bsize is created.

OSSL_PARAM_construct_octet_ptr() is a function that constructs an OCTET string pointer OSSL_PARAM structure. A parameter with name key, storage pointer *buf and size bsize is created.

OSSL_PARAM_construct_end() is a function that constructs the terminating OSSL_PARAM structure.

OSSL_PARAM_locate() is a function that searches an array of parameters for the one matching the key name.

OSSL_PARAM_locate_const() behaves exactly like OSSL_PARAM_locate() except for the presence of const for the array argument and its return value.

OSSL_PARAM_get_TYPE() retrieves a value of type TYPE from the parameter p. The value is copied to the address val. Type coercion takes place as discussed in the NOTES section.

OSSL_PARAM_set_TYPE() stores a value val of type TYPE into the parameter p. If the parameter's data field is NULL, then only its return_size field will be assigned the size the parameter's data buffer should have. Type coercion takes place as discussed in the NOTES section.

OSSL_PARAM_get_BN() retrieves a BIGNUM from the parameter pointed to by p. The BIGNUM referenced by val is updated and is allocated if *val is NULL.

OSSL_PARAM_set_BN() stores the BIGNUM val into the parameter p. If the parameter's data field is NULL, then only its return_size field will be assigned the size the parameter's data buffer should have.

OSSL_PARAM_get_utf8_string() retrieves a UTF8 string from the parameter

pointed to by p. The string is stored into *val with a size limit of max_len, which must be large enough to accomodate a terminating NUL byte, otherwise this function will fail. If *val is NULL, memory is allocated for the string (including the terminating NUL byte) and max_len is ignored. If memory is allocated by this function, it must be freed by the caller.

OSSL_PARAM_set_utf8_string() sets a UTF8 string from the parameter pointed to by p to the value referenced by val. If the parameter's data field isn't NULL, its data_size must indicate that the buffer is large enough to accomodate the string that val points at, not including the terminating NUL byte, or this function will fail. A terminating NUL byte is added only if the parameter's data_size indicates the buffer is longer than the string length, otherwise the string will not be NUL terminated. If the parameter's data field is NULL, then only its return_size field will be assigned the minimum size the parameter's data buffer should have to accomodate the string, not including a terminating NUL byte.

OSSL_PARAM_get_octet_string() retrieves an OCTET string from the parameter pointed to by p. The OCTETs are either stored into *val with a length limit of max_len or, in the case when *val is NULL, memory is allocated and max_len is ignored. *used_len is populated with the number of OCTETs stored. If val is NULL then the OCTETS are not stored, but *used_len is still populated. If memory is allocated by this function, it must be freed by the caller.

OSSL_PARAM_set_octet_string() sets an OCTET string from the parameter pointed to by p to the value referenced by val. If the parameter's data field is NULL, then only its return_size field will be assigned the size the parameter's data buffer should have.

OSSL_PARAM_get_utf8_ptr() retrieves the UTF8 string pointer from the

parameter referenced by p and stores it in *val.

OSSL_PARAM_set_utf8_ptr() sets the UTF8 string pointer in the parameter referenced by p to the values val.

OSSL_PARAM_get_octet_ptr() retrieves the OCTET string pointer from the parameter referenced by p and stores it in *val. The length of the OCTET string is stored in *used_len.

OSSL_PARAM_set_octet_ptr() sets the OCTET string pointer in the parameter referenced by p to the values val. The length of the OCTET string is provided by used_len.

OSSL_PARAM_get_utf8_string_ptr() retrieves the pointer to a UTF8 string from the parameter pointed to by p, and stores that pointer in *val. This is different from OSSL_PARAM_get_utf8_string(), which copies the string.

OSSL_PARAM_get_octet_string_ptr() retrieves the pointer to a octet string from the parameter pointed to by p, and stores that pointer in *val, along with the string's length in *used_len. This is different from OSSL_PARAM_get_octet_string(), which copies the string.

The OSSL_PARAM_UNMODIFIED macro is used to detect if a parameter was set. On creation, via either the macros or construct calls, the return_size field is set to this. If the parameter is set using the calls defined herein, the return_size field is changed.

OSSL_PARAM_modified() queries if the parameter param has been set or not using the calls defined herein.

OSSL_PARAM_set_all_unmodified() resets the unused indicator for all parameters in the array params.

RETURN VALUES

OSSL_PARAM_construct_TYPE(), OSSL_PARAM_construct_BN(), OSSL_PARAM_construct_utf8_string(), OSSL_PARAM_construct_octet_string(), OSSL_PARAM_construct_utf8_ptr() and OSSL_PARAM_construct_octet_ptr() return a populated OSSL_PARAM structure.

OSSL_PARAM_locate() and OSSL_PARAM_locate_const() return a pointer to the matching OSSL_PARAM object. They return NULL on error or when no object matching key exists in the array.

OSSL_PARAM_modified() returns 1 if the parameter was set and 0 otherwise.

All other functions return 1 on success and 0 on failure.

NOTES

Native types will be converted as required only if the value is exactly representable by the target type or parameter. Apart from that, the functions must be used appropriately for the expected type of the parameter.

OSSL_PARAM_get_BN() and OSSL_PARAM_set_BN() currently only support nonnegative BIGNUMs, and by consequence, only OSSL_PARAM_UNSIGNED_INTEGER. OSSL_PARAM_construct_BN() currently constructs an OSSL_PARAM structure with the data type OSSL_PARAM_UNSIGNED_INTEGER.

For OSSL_PARAM_construct_utf8_ptr() and OSSL_PARAM_consstruct_octet_ptr(), bsize is not relevant if the purpose is to send the OSSL_PARAM array to a responder, i.e. to get parameter data back. In that case, bsize can safely be given zero. See

"DESCRIPTION" in OSSL_PARAM(3) for further information on the possible purposes.

EXAMPLES

Reusing the examples from OSSL_PARAM(3) to just show how OSSL_PARAM arrays can be handled using the macros and functions defined herein.

Example 1

This example is for setting parameters on some object:

```
#include <openssl/core.h>

const char *foo = "some string";
size_t foo_l = strlen(foo);
const char bar[] = "some other string";
const OSSL_PARAM set[] = {
    OSSL_PARAM_utf8_ptr("foo", &foo, foo_l),
    OSSL_PARAM_utf8_string("bar", bar, sizeof(bar) - 1),
    OSSL_PARAM_END
};
```

Example 2

This example is for requesting parameters on some object, and also demonstrates that the requestor isn't obligated to request all available parameters:

```
const char *foo = NULL;
char bar[1024];
OSSL_PARAM request[] = {
    OSSL_PARAM_utf8_ptr("foo", &foo, 0),
    OSSL_PARAM_utf8_string("bar", bar, sizeof(bar)),
    OSSL_PARAM_END
};
```

A responder that receives this array (as "params" in this example)
could fill in the parameters like this:

```
/* OSSL_PARAM *params */

OSSL_PARAM *p;

if ((p = OSSL_PARAM_locate(params, "foo")) != NULL)
    OSSL_PARAM_set_utf8_ptr(p, "foo value");
if ((p = OSSL_PARAM_locate(params, "bar")) != NULL)
    OSSL_PARAM_set_utf8_string(p, "bar value");
if ((p = OSSL_PARAM_locate(params, "cookie")) != NULL)
    OSSL_PARAM_set_utf8_ptr(p, "cookie value");
```

SEE ALSO

openssl-core.h(7), OSSL_PARAM(3)

HISTORY

These APIs were introduced in OpenSSL 3.0.

COPYRIGHT