

**NAME**

OSSL\_PARAM\_BLD, OSSL\_PARAM\_BLD\_new, OSSL\_PARAM\_BLD\_to\_param,  
 OSSL\_PARAM\_BLD\_free, OSSL\_PARAM\_BLD\_push\_int, OSSL\_PARAM\_BLD\_push\_uint,  
 OSSL\_PARAM\_BLD\_push\_long, OSSL\_PARAM\_BLD\_push\_ulong,  
 OSSL\_PARAM\_BLD\_push\_int32, OSSL\_PARAM\_BLD\_push\_uint32,  
 OSSL\_PARAM\_BLD\_push\_int64, OSSL\_PARAM\_BLD\_push\_uint64,  
 OSSL\_PARAM\_BLD\_push\_size\_t, OSSL\_PARAM\_BLD\_push\_time\_t,  
 OSSL\_PARAM\_BLD\_push\_double, OSSL\_PARAM\_BLD\_push\_BN,  
 OSSL\_PARAM\_BLD\_push\_BN\_pad, OSSL\_PARAM\_BLD\_push\_utf8\_string,  
 OSSL\_PARAM\_BLD\_push\_utf8\_ptr, OSSL\_PARAM\_BLD\_push\_octet\_string,  
 OSSL\_PARAM\_BLD\_push\_octet\_ptr - functions to assist in the creation of OSSL\_PARAM arrays

**SYNOPSIS**

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

typedef struct OSSL_PARAM_BLD;

OSSL_PARAM_BLD *OSSL_PARAM_BLD_new(void);
OSSL_PARAM *OSSL_PARAM_BLD_to_param(OSSL_PARAM_BLD *bld);
void OSSL_PARAM_BLD_free(OSSL_PARAM_BLD *bld);

int OSSL_PARAM_BLD_push_TYPE(OSSL_PARAM_BLD *bld, const char *key, TYPE val);

int OSSL_PARAM_BLD_push_BN(OSSL_PARAM_BLD *bld, const char *key,
                           const BIGNUM *bn);
int OSSL_PARAM_BLD_push_BN_pad(OSSL_PARAM_BLD *bld, const char *key,
                               const BIGNUM *bn, size_t sz);

int OSSL_PARAM_BLD_push_utf8_string(OSSL_PARAM_BLD *bld, const char *key,
                                    const char *buf, size_t bsize);
int OSSL_PARAM_BLD_push_utf8_ptr(OSSL_PARAM_BLD *bld, const char *key,
                                  char *buf, size_t bsize);
int OSSL_PARAM_BLD_push_octet_string(OSSL_PARAM_BLD *bld, const char *key,
                                     const void *buf, size_t bsize);
int OSSL_PARAM_BLD_push_octet_ptr(OSSL_PARAM_BLD *bld, const char *key,
                                   void *buf, size_t bsize);
```

**DESCRIPTION**

A collection of utility functions that simplify the creation of OSSL\_PARAM arrays. The *TYPE* names are as per **OSSL\_PARAM\_int(3)**.

**OSSL\_PARAM\_BLD\_new()** allocates and initialises a new **OSSL\_PARAM\_BLD** structure so that values can be added. Any existing values are cleared.

**OSSL\_PARAM\_BLD\_free()** deallocates the memory allocated by **OSSL\_PARAM\_BLD\_new()**.

**OSSL\_PARAM\_BLD\_to\_param()** converts a built up **OSSL\_PARAM\_BLD** structure *bld* into an allocated **OSSL\_PARAM** array. The **OSSL\_PARAM** array and all associated storage must be freed by calling **OSSL\_PARAM\_free()** with the functions return value. **OSSL\_PARAM\_BLD\_free()** can safely be called any time after this function is.

**OSSL\_PARAM\_BLD\_push\_TYPE()** are a series of functions which will create **OSSL\_PARAM** objects of the specified size and correct type for the *val* argument. *val* is stored by value and an expression or auto variable can be used.

**OSSL\_PARAM\_BLD\_push\_BN()** is a function that will create an **OSSL\_PARAM** object that holds the specified **BIGNUM** *bn*. If *bn* is marked as being securely allocated, its **OSSL\_PARAM** representation will also be securely allocated. The *bn* argument is stored by reference and the underlying **BIGNUM** object must exist until after **OSSL\_PARAM\_BLD\_to\_param()** has been called.

**OSSL\_PARAM\_BLD\_push\_BN\_pad()** is a function that will create an **OSSL\_PARAM** object that holds the specified **BIGNUM** *bn*. The object will be padded to occupy exactly *sz* bytes, if insufficient space is specified an error results. If *bn* is marked as being securely allocated, its **OSSL\_PARAM** representation will also be securely allocated. The *bn* argument is stored by reference and the underlying **BIGNUM** object must exist until after **OSSL\_PARAM\_BLD\_to\_param()** has been called.

**OSSL\_PARAM\_BLD\_push\_utf8\_string()** is a function that will create an **OSSL\_PARAM** object that references the UTF8 string specified by *buf*. The length of the string *bsize* should not include the terminating NUL byte. If it is zero then it will be calculated. The string that *buf* points to is stored by reference and must remain in scope until after **OSSL\_PARAM\_BLD\_to\_param()** has been called.

**OSSL\_PARAM\_BLD\_push\_octet\_string()** is a function that will create an **OSSL\_PARAM** object that references the octet string specified by *buf* and *<bsize>*. The memory that *buf* points to is stored by reference and must remain in scope until after **OSSL\_PARAM\_BLD\_to\_param()** has been called.

**OSSL\_PARAM\_BLD\_push\_utf8\_ptr()** is a function that will create an **OSSL\_PARAM** object that references the UTF8 string specified by *buf*. The length of the string *bsize* should not include the terminating NUL byte. If it is zero then it will be calculated. The string *buf* points to is stored by reference and must remain in scope until the **OSSL\_PARAM** array is freed.

**OSSL\_PARAM\_BLD\_push\_octet\_ptr()** is a function that will create an **OSSL\_PARAM** object that

references the octet string specified by *buf*. The memory *buf* points to is stored by reference and must remain in scope until the OSSL\_PARAM array is freed.

## RETURN VALUES

**OSSL\_PARAM\_BLD\_new()** returns the allocated OSSL\_PARAM\_BLD structure, or NULL on error.

**OSSL\_PARAM\_BLD\_to\_param()** returns the allocated OSSL\_PARAM array, or NULL on error.

All of the OSSL\_PARAM\_BLD\_push\_TYPE functions return 1 on success and 0 on error.

## NOTES

**OSSL\_PARAM\_BLD\_push\_BN()** and **OSSL\_PARAM\_BLD\_push\_BN\_pad()** currently only support nonnegative **BIGNUM**s. They return an error on negative **BIGNUM**s.

## EXAMPLES

Both examples creating an OSSL\_PARAM array that contains an RSA key. For both, the predefined key variables are:

```
BIGNUM *n;      /* modulus */
unsigned int e; /* public exponent */
BIGNUM *d;      /* private exponent */
BIGNUM *p, *q;  /* first two prime factors */
BIGNUM *dmp1, *dmq1; /* first two CRT exponents */
BIGNUM *iqmp;   /* first CRT coefficient */
```

### Example 1

This example shows how to create an OSSL\_PARAM array that contains an RSA private key.

```
OSSL_PARAM_BLD *bld = OSSL_PARAM_BLD_new();
OSSL_PARAM *params = NULL;

if (bld == NULL)
    || !OSSL_PARAM_BLD_push_BN(bld, "n", n)
    || !OSSL_PARAM_BLD_push_uint(bld, "e", e)
    || !OSSL_PARAM_BLD_push_BN(bld, "d", d)
    || !OSSL_PARAM_BLD_push_BN(bld, "rsa-factor1", p)
    || !OSSL_PARAM_BLD_push_BN(bld, "rsa-factor2", q)
    || !OSSL_PARAM_BLD_push_BN(bld, "rsa-exponent1", dmp1)
    || !OSSL_PARAM_BLD_push_BN(bld, "rsa-exponent2", dmq1)
    || !OSSL_PARAM_BLD_push_BN(bld, "rsa-coefficient1", iqmp)
```

```
    || (params = OSSL_PARAM_BLD_to_param(bld)) == NULL)
    goto err;
OSSL_PARAM_BLD_free(bld);
/* Use params */
...
OSSL_PARAM_free(params);
```

### Example 2

This example shows how to create an OSSL\_PARAM array that contains an RSA public key.

```
OSSL_PARAM_BLD *bld = OSSL_PARAM_BLD_new();
OSSL_PARAM *params = NULL;

if (nld == NULL
    || !OSSL_PARAM_BLD_push_BN(bld, "n", n)
    || !OSSL_PARAM_BLD_push_uint(bld, "e", e)
    || (params = OSSL_PARAM_BLD_to_param(bld)) == NULL)
    goto err;
OSSL_PARAM_BLD_free(bld);
/* Use params */
...
OSSL_PARAM_free(params);
```

### SEE ALSO

**OSSL\_PARAM\_int(3)**, **OSSL\_PARAM(3)**, **OSSL\_PARAM\_free(3)**

### HISTORY

The functions described here were all added in OpenSSL 3.0.

### COPYRIGHT

Copyright 2019-2022 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or at <<https://www.openssl.org/source/license.html>>.