

**NAME**

ASN1\_EXTERN\_FUNCS, ASN1\_ex\_d2i, ASN1\_ex\_d2i\_ex, ASN1\_ex\_i2d, ASN1\_ex\_new\_func, ASN1\_ex\_new\_ex\_func, ASN1\_ex\_free\_func, ASN1\_ex\_print\_func, IMPLEMENT\_EXTERN\_ASN1  
 - ASN.1 external function support

**SYNOPSIS**

```
#include <openssl/asn1t.h>
```

```
typedef int ASN1_ex_d2i(ASN1_VALUE **pval, const unsigned char **in, long len,
    const ASN1_ITEM *it, int tag, int aclass, char opt,
    ASN1_TLC *ctx);
```

```
typedef int ASN1_ex_d2i_ex(ASN1_VALUE **pval, const unsigned char **in, long len,
    const ASN1_ITEM *it, int tag, int aclass, char opt,
    ASN1_TLC *ctx, OSSL_LIB_CTX *libctx,
    const char *propq);
```

```
typedef int ASN1_ex_i2d(const ASN1_VALUE **pval, unsigned char **out,
    const ASN1_ITEM *it, int tag, int aclass);
```

```
typedef int ASN1_ex_new_func(ASN1_VALUE **pval, const ASN1_ITEM *it);
```

```
typedef int ASN1_ex_new_ex_func(ASN1_VALUE **pval, const ASN1_ITEM *it,
    OSSL_LIB_CTX *libctx, const char *propq);
```

```
typedef void ASN1_ex_free_func(ASN1_VALUE **pval, const ASN1_ITEM *it);
```

```
typedef int ASN1_ex_print_func(BIO *out, const ASN1_VALUE **pval,
    int indent, const char *fname,
    const ASN1_PCTX *pctx);
```

```
struct ASN1_EXTERN_FUNCS_st {
    void *app_data;
    ASN1_ex_new_func *asn1_ex_new;
    ASN1_ex_free_func *asn1_ex_free;
    ASN1_ex_free_func *asn1_ex_clear;
    ASN1_ex_d2i *asn1_ex_d2i;
    ASN1_ex_i2d *asn1_ex_i2d;
    ASN1_ex_print_func *asn1_ex_print;
    ASN1_ex_new_ex_func *asn1_ex_new_ex;
    ASN1_ex_d2i_ex *asn1_ex_d2i_ex;
};
typedef struct ASN1_EXTERN_FUNCS_st ASN1_EXTERN_FUNCS;
```

```
#define IMPLEMENT_EXTERN_ASN1(sname, tag, fptrs)
```

## DESCRIPTION

ASN.1 data structures templates are typically defined in OpenSSL using a series of macros such as **ASN1\_SEQUENCE()**, **ASN1\_SEQUENCE\_END()** and so on. Instead templates can also be defined based entirely on external functions. These external functions are called to perform operations such as creating a new **ASN1\_VALUE** or converting an **ASN1\_VALUE** to or from DER encoding.

The macro **IMPLEMENT\_EXTERN\_ASN1()** can be used to create such an externally defined structure. The name of the structure should be supplied in the *sname* parameter. The tag for the structure (e.g. typically **V\_ASN1\_SEQUENCE**) should be supplied in the *tag* parameter. Finally a pointer to an **ASN1\_EXTERN\_FUNCS** structure should be supplied in the *fptrs* parameter.

The **ASN1\_EXTERN\_FUNCS** structure has the following entries.

### *app\_data*

A pointer to arbitrary application specific data.

### *asn1\_ex\_new*

A "new" function responsible for constructing a new **ASN1\_VALUE** object. The newly constructed value should be stored in *\*pval*. The *it* parameter is a pointer to the **ASN1\_ITEM** template object created via the **IMPLEMENT\_EXTERN\_ASN1()** macro.

Returns a positive value on success or 0 on error.

### *asn1\_ex\_free*

A "free" function responsible for freeing the **ASN1\_VALUE** passed in *\*pval* that was previously allocated via a "new" function. The *it* parameter is a pointer to the **ASN1\_ITEM** template object created via the **IMPLEMENT\_EXTERN\_ASN1()** macro.

### *asn1\_ex\_clear*

A "clear" function responsible for clearing any data in the **ASN1\_VALUE** passed in *\*pval* and making it suitable for reuse. The *it* parameter is a pointer to the **ASN1\_ITEM** template object created via the **IMPLEMENT\_EXTERN\_ASN1()** macro.

### *asn1\_ex\_d2i*

A "d2i" function responsible for converting DER data with the tag *tag* and class *class* into an **ASN1\_VALUE**. If *\*pval* is non-NULL then the **ASN1\_VALUE** it points to should be reused. Otherwise a new **ASN1\_VALUE** should be allocated and stored in *\*pval*. *\*in* points to the DER data to be decoded and *len* is the length of that data. After decoding *\*in* should be updated to point at the next byte after the decoded data. If the **ASN1\_VALUE** is considered optional in this context then *opt* will be nonzero. Otherwise it will be zero. The *it* parameter is a pointer to the

**ASN1\_ITEM** template object created via the **IMPLEMENT\_EXTERN\_ASN1()** macro. A pointer to the current **ASN1\_TLC** context (which may be required for other ASN1 function calls) is passed in the *ctx* parameter.

The *asn1\_ex\_d2i* entry may be NULL if *asn1\_ex\_d2i\_ex* has been specified instead.

Returns  $\leq 0$  on error or a positive value on success.

#### *asn1\_ex\_i2d*

An "i2d" function responsible for converting an **ASN1\_VALUE** into DER encoding. On entry *\*pval* will contain the **ASN1\_VALUE** to be encoded. If default tagging is to be used then *tag* will be -1 on entry. Otherwise if implicit tagging should be used then *tag* and *aclass* will be the tag and associated class.

If *out* is not NULL then this function should write the DER encoded data to the buffer in *\*out*, and then increment *\*out* to point to immediately after the data just written.

If *out* is NULL then no data should be written but the length calculated and returned as if it were.

The *asn1\_ex\_i2d* entry may be NULL if *asn1\_ex\_i2d\_ex* has been specified instead.

The return value should be negative if a fatal error occurred, or 0 if a non-fatal error occurred. Otherwise it should return the length of the encoded data.

#### *asn1\_ex\_print*

A "print" function. *out* is the BIO to print the output to. *\*pval* is the **ASN1\_VALUE** to be printed. *indent* is the number of spaces of indenting to be printed before any data is printed. *fname* is currently unused and is always "". *pctx* is a pointer to the **ASN1\_PCTX** for the print operation.

Returns 0 on error or a positive value on success. If the return value is 2 then an additional newline will be printed after the data printed by this function.

#### *asn1\_ex\_new\_ex*

This is the same as *asn1\_ex\_new* except that it is additionally passed the **OSSL\_LIB\_CTX** to be used in *libctx* and any property query string to be used for algorithm fetching in the *propq* parameter. See "ALGORITHM FETCHING" in **crypto(7)** for further details. If *asn1\_ex\_new\_ex* is non NULL, then it will always be called in preference to *asn1\_ex\_new*.

#### *asn1\_ex\_d2i\_ex*

This is the same as *asn1\_ex\_d2i* except that it is additionally passed the **OSSL\_LIB\_CTX** to be

used in *libctx* and any property query string to be used for algorithm fetching in the *propq* parameter. See "ALGORITHM FETCHING" in **crypto**(7) for further details. If *asn1\_ex\_d2i\_ex* is non NULL, then it will always be called in preference to *asn1\_ex\_d2i*.

## RETURN VALUES

Return values for the various callbacks are as described above.

## SEE ALSO

**ASN1\_item\_new\_ex**(3)

## HISTORY

The *asn1\_ex\_new\_ex* and *asn1\_ex\_d2i\_ex* callbacks were added in OpenSSL 3.0.

## COPYRIGHT

Copyright 2021 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>.