NAME

ASN1_TYPE_get, ASN1_TYPE_set, ASN1_TYPE_set1, ASN1_TYPE_cmp, ASN1_TYPE_unpack_sequence, ASN1_TYPE_pack_sequence - ASN1_TYPE utility functions

SYNOPSIS

```
#include <openssl/asn1.h>

int ASN1_TYPE_get(const ASN1_TYPE *a);
void ASN1_TYPE_set(ASN1_TYPE *a, int type, void *value);
int ASN1_TYPE_set1(ASN1_TYPE *a, int type, const void *value);
int ASN1_TYPE_cmp(const ASN1_TYPE *a, const ASN1_TYPE *b);

void *ASN1_TYPE_unpack_sequence(const ASN1_ITEM *it, const ASN1_TYPE *t);
ASN1_TYPE *ASN1_TYPE_pack_sequence(const ASN1_ITEM *it, void *s,
                                   ASN1_TYPE **t);
```

DESCRIPTION

These functions allow an **ASN1_TYPE** structure to be manipulated. The **ASN1_TYPE** structure can contain any ASN.1 type or constructed type such as a SEQUENCE: it is effectively equivalent to the ASN.1 ANY type.

ASN1_TYPE_get() returns the type of *a* or 0 if it fails.

ASN1_TYPE_set() sets the value of *a* to *type* and *value*. This function uses the pointer *value* internally so it must **not** be freed up after the call.

ASN1_TYPE_set1() sets the value of *a* to *type* a copy of *value*.

ASN1_TYPE_cmp() compares ASN.1 types *a* and *b* and returns 0 if they are identical and nonzero otherwise.

ASN1_TYPE_unpack_sequence() attempts to parse the SEQUENCE present in *t* using the ASN.1 structure *it*. If successful it returns a pointer to the ASN.1 structure corresponding to *it* which must be freed by the caller. If it fails it return NULL.

ASN1_TYPE_pack_sequence() attempts to encode the ASN.1 structure *s* corresponding to *it* into an **ASN1_TYPE**. If successful the encoded **ASN1_TYPE** is returned. If *t* and **t* are not NULL the encoded type is written to *t* overwriting any existing data. If *t* is not NULL but **t* is NULL the returned **ASN1_TYPE** is written to **t*.

NOTES

The type and meaning of the *value* parameter for **ASN1_TYPE_set()** and **ASN1_TYPE_set1()** is determined by the *type* parameter. If *type* is **V_ASN1_NULL** *value* is ignored. If *type* is **V_ASN1_BOOLEAN** then the boolean is set to TRUE if *value* is not NULL. If *type* is **V_ASN1_OBJECT** then value is an **ASN1_OBJECT** structure. Otherwise *type* is and **ASN1_STRING** structure. If *type* corresponds to a primitive type (or a string type) then the contents of the **ASN1_STRING** contain the content octets of the type. If *type* corresponds to a constructed type or a tagged type (**V_ASN1_SEQUENCE**, **V_ASN1_SET** or **V_ASN1_OTHER**) then the **ASN1_STRING** contains the entire ASN.1 encoding verbatim (including tag and length octets).

ASN1_TYPE_cmp() may not return zero if two types are equivalent but have different encodings. For example the single content octet of the boolean TRUE value under BER can have any nonzero encoding but **ASN1_TYPE_cmp()** will only return zero if the values are the same.

If either or both of the parameters passed to **ASN1_TYPE_cmp()** is NULL the return value is nonzero. Technically if both parameters are NULL the two types could be absent OPTIONAL fields and so should match, however, passing NULL values could also indicate a programming error (for example an unparsable type which returns NULL) for types which do **not** match. So applications should handle the case of two absent values separately.

RETURN VALUES

ASN1_TYPE_get() returns the type of the **ASN1_TYPE** argument.

ASN1_TYPE_set() does not return a value.

ASN1_TYPE_set1() returns 1 for success and 0 for failure.

ASN1_TYPE_cmp() returns 0 if the types are identical and nonzero otherwise.

ASN1_TYPE_unpack_sequence() returns a pointer to an ASN.1 structure or NULL on failure.

ASN1_TYPE_pack_sequence() return an **ASN1_TYPE** structure if it succeeds or NULL on failure.

COPYRIGHT