NAME

ASN1_STRING_print_ex, ASN1_STRING_print_ex_fp, ASN1_STRING_print - ASN1_STRING output routines.

SYNOPSIS

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

int ASN1_STRING_print_ex(BIO *out, ASN1_STRING *str, unsigned long flags);
int ASN1_STRING_print_ex_fp(FILE *fp, ASN1_STRING *str, unsigned long flags);
int ASN1_STRING_print(BIO *out, ASN1_STRING *str);
```

DESCRIPTION

These functions output an **ASN1_STRING** structure. **ASN1_STRING** is used to represent all the ASN1 string types.

ASN1_STRING_print_ex() outputs **str** to **out**, the format is determined by the options **flags**. *ASN1_STRING_print_ex_fp()* is identical except it outputs to **fp** instead.

ASN1_STRING_print() prints **str** to **out** but using a different format to *ASN1_STRING_print_ex()*. It replaces unprintable characters (other than CR, LF) with '.'.

NOTES

ASN1_STRING_print() is a legacy function which should be avoided in new applications.

Although there are a large number of options frequently **ASN1_STRFLGS_RFC2253** is suitable, or on UTF8 terminals **ASN1_STRFLGS_RFC2253 & ~ASN1_STRFLGS_ESC_MSB**.

The complete set of supported options for **flags** is listed below.

Various characters can be escaped. If **ASN1_STRFLGS_ESC_2253** is set the characters determined by RFC2253 are escaped. If **ASN1_STRFLGS_ESC_CTRL** is set control characters are escaped. If **ASN1_STRFLGS_ESC_MSB** is set characters with the MSB set are escaped: this option should **not** be used if the terminal correctly interprets UTF8 sequences.

Escaping takes several forms.

If the character being escaped is a 16 bit character then the form "UXXXX" is used using exactly four characters for the hex representation. If it is 32 bits then "WXXXXXXXX" is used using eight characters of its hex representation. These forms will only be used if UTF8 conversion is not set (see below).

Printable characters are normally escaped using the backslash '' character. If **ASN1_STRFLGS_ESC_QUOTE** is set then the whole string is instead surrounded by double quote characters: this is arguably more readable than the backslash notation. Other characters use the "XX" using exactly two characters of the hex representation.

If **ASN1_STRFLGS_UTF8_CONVERT** is set then characters are converted to UTF8 format first. If the terminal supports the display of UTF8 sequences then this option will correctly display multi byte characters.

If **ASN1_STRFLGS_IGNORE_TYPE** is set then the string type is not interpreted at all: everything is assumed to be one byte per character. This is primarily for debugging purposes and can result in confusing output in multi character strings.

If **ASN1_STRFLGS_SHOW_TYPE** is set then the string type itself is printed out before its value (for example "BMPSTRING"), this actually uses *ASN1_tag2str()*.

The content of a string instead of being interpreted can be "dumped": this just outputs the value of the string using the form #XXXX using hex format for each octet.

If **ASN1_STRFLGS_DUMP_ALL** is set then any type is dumped.

Normally non character string types (such as OCTET STRING) are assumed to be one byte per character, if **ASN1_STRFLGS_DUMP_UNKNOWN** is set then they will be dumped instead.

When a type is dumped normally just the content octets are printed, if **ASN1_STRFLGS_DUMP_DER** is set then the complete encoding is dumped instead (including tag and length octets).

ASN1_STRFLGS_RFC2253 includes all the flags required by RFC2253. It is equivalent to: ASN1_STRFLGS_ESC_2253 | ASN1_STRFLGS_ESC_CTRL | ASN1_STRFLGS_ESC_MSB | ASN1_STRFLGS_UTF8_CONVERT | ASN1_STRFLGS_DUMP_UNKNOWN ASN1_STRFLGS_DUMP_DER

SEE ALSO

X509_NAME_print_ex(3), *ASN1_tag2str(3)*

HISTORY

TBA