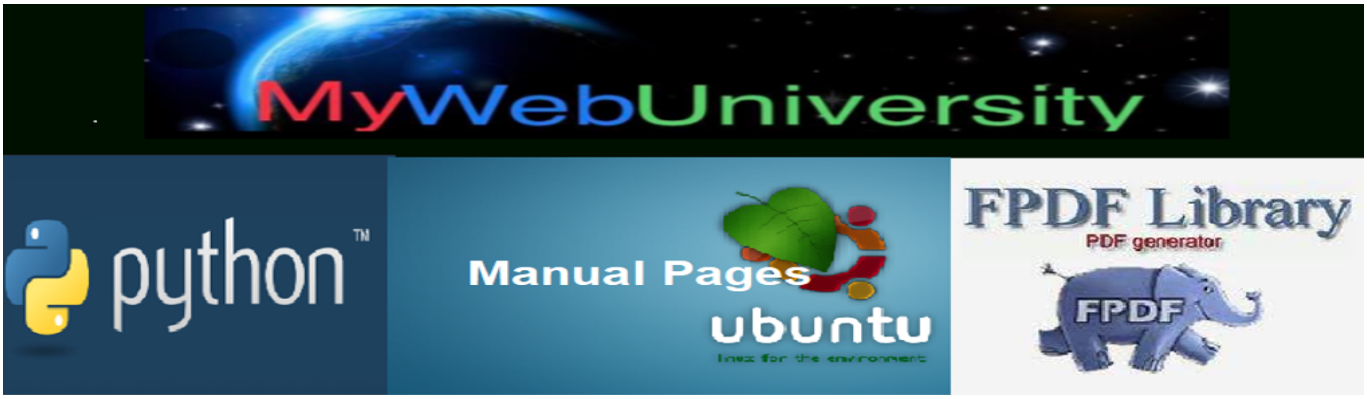

Rocky Enterprise Linux 9.2 Manual Pages on command 'openssl-core.h.7ssl'

$ man openssl-core.h.7ssl

OPENSSL-CORE.H(7SSL) OpenSSL OPENSSL-CORE.H(7SSL)

NAME

openssl/core.h - OpenSSL Core types

SYNOPSIS

#include <openssl/core.h>

DESCRIPTION

The <openssl/core.h> header defines a number of public types that are used to communicate between the OpenSSL libraries and implementation providers. These types are designed to minimise the need for intimate knowledge of internal structures between the OpenSSL libraries and the providers.

The types are:

OSSL_DISPATCH

This type is a tuple of function identity and function pointer. Arrays of this type are passed between the OpenSSL libraries and the providers to describe what functionality one side provides to the other. Arrays of this type must be terminated with a tuple having function identity zero and function pointer NULL.

The available function identities and corresponding function signatures are defined in openssl-core_dispatch.h(7).

Any function identity not recognised by the recipient of this type will be ignored.

This ensures that providers built with one OpenSSL version in mind will work together with any other OpenSSL version that supports this mechanism.

OSSL_ITEM

This type is a tuple of integer and pointer. It's a generic type used as a generic

descriptor, its exact meaning being defined by how it's used. Arrays of this type are passed between the OpenSSL libraries and the providers, and must be terminated with a tuple where the integer is zero and the pointer NULL.

OSSL_ALGORITHM

This type is a tuple of an algorithm name (string), a property definition (string) and a dispatch table (array of OSSL_DISPATCH). Arrays of this type are passed on demand from the providers to the OpenSSL libraries to describe what algorithms the providers provide implementations of, and with what properties. Arrays of this type must be terminated with a tuple having function identity zero and function pointer NULL.

The algorithm names and property definitions are defined by the providers.

The OpenSSL libraries use the first of the algorithm names as the main or canonical name, on a per algorithm implementation basis.

OSSL_PARAM

This type is a structure that allows passing arbitrary object data between two parties that have no or very little shared knowledge about their respective internal structures for that object. It's normally passed in arrays, where the array is terminated with an element where all fields are zero (for non-pointers) or NULL (for pointers).

These arrays can be used to set parameters for some object, to request parameters, and to describe parameters.

OSSL_PARAM is further described in OSSL_PARAM(3)

OSSL_CALLBACK

This is a function type for a generic feedback callback function:

```
typedef int (OSSL_CALLBACK)(const OSSL_PARAM params[], void *arg);
```

A function that takes a pointer of this type should also take a pointer to caller data. When calling this callback, the function is expected to build an OSSL_PARAM array of data it wants or is expected to pass back, and pass that as params, as well as the caller data pointer it received, as arg.

OSSL_PASSPHRASE_CALLBACK

This is a function type for a generic pass phrase callback function:

```
typedef int (OSSL_PASSPHRASE_CALLBACK)(char *pass, size_t pass_size,
                                       size_t *pass_len,
                                       const OSSL_PARAM params[],
```

void *arg);

This callback can be used to prompt the user for a passphrase. When calling it, a buffer to store the pass phrase needs to be given with pass, and its size with pass_size. The length of the prompted pass phrase will be given back in *pass_len.

Additional parameters can be passed with the OSSL_PARAM array params.

A function that takes a pointer of this type should also take a pointer to caller data, which should be passed as arg to this callback.

SEE ALSO

openssl-core_dispatch.h(7)

HISTORY

The types described here were added in OpenSSL 3.0.

COPYRIGHT