



Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'provider.7ssl'

\$ man provider.7ssl

PROVIDER(7SSL) OpenSSL PROVIDER(7SSL)

NAME

provider - OpenSSL operation implementation providers

SYNOPSIS

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

DESCRIPTION

General

This page contains information useful to provider authors.

A provider, in OpenSSL terms, is a unit of code that provides one or more implementations for various operations for diverse algorithms that one might want to perform.

An operation is something one wants to do, such as encryption and decryption, key derivation, MAC calculation, signing and verification, etc.

An algorithm is a named method to perform an operation. Very often, the algorithms revolve around cryptographic operations, but may also revolve around other types of operation, such as managing certain types of objects.

See crypto(7) for further details.

Provider

A provider offers an initialization function, as a set of base functions in the form of an OSSL_DISPATCH array, and by extension, a set of OSSL_ALGORITHMs (see openssl-core.h(7)).

It may be a dynamically loadable module, or may be built-in, in OpenSSL libraries or in the application. If it's a dynamically loadable module, the initialization function must be named "OSSL_provider_init" and must be exported. If it's built-in, the initialization function may have any name.

The initialization function must have the following signature:

```
int NAME(const OSSL_CORE_HANDLE *handle,  
         const OSSL_DISPATCH *in, const OSSL_DISPATCH **out,  
         void **provctx);
```

handle is the OpenSSL library object for the provider, and works as a handle for everything the OpenSSL libraries need to know about the provider. For the provider itself, it is passed to some of the functions given in the dispatch array in.

in is a dispatch array of base functions offered by the OpenSSL libraries, and the available functions are further described in provider-base(7).

*out must be assigned a dispatch array of base functions that the provider offers to the OpenSSL libraries. The functions that may be offered are further described in provider-base(7), and they are the central means of communication between the OpenSSL libraries and the provider.

*provctx should be assigned a provider specific context to allow the provider multiple simultaneous uses. This pointer will be passed to various operation functions offered by the provider.

Note that the provider will not be made available for applications to use until the initialization function has completed and returned successfully.

One of the functions the provider offers to the OpenSSL libraries is the central mechanism for the OpenSSL libraries to get access to operation implementations for diverse algorithms. Its referred to with the number OSSL_FUNC_PROVIDER_QUERY_OPERATION and has the following signature:

```
const OSSL_ALGORITHM *provider_query_operation(void *provctx,  
                                              int operation_id,  
                                              const int *no_store);
```

provctx is the provider specific context that was passed back by the initialization function.

operation_id is an operation identity (see "Operations" below).

no_store is a flag back to the OpenSSL libraries which, when nonzero, signifies that the OpenSSL libraries will not store a reference to the returned data in their internal store of implementations.

The returned OSSL_ALGORITHM is the foundation of any OpenSSL library API that uses providers for their implementation, most commonly in the fetching type of functions (see

"ALGORITHM FETCHING" in `crypto(7)`).

Operations

Operations are referred to with numbers, via macros with names starting with "OSSL_OP_".

With each operation comes a set of defined function types that a provider may or may not offer, depending on its needs.

Currently available operations are:

Digests

In the OpenSSL libraries, the corresponding method object is `EVP_MD`. The number for this operation is `OSSL_OP_DIGEST`. The functions the provider can offer are described in `provider-digest(7)`

Symmetric ciphers

In the OpenSSL libraries, the corresponding method object is `EVP_CIPHER`. The number for this operation is `OSSL_OP_CIPHER`. The functions the provider can offer are described in `provider-cipher(7)`

Message Authentication Code (MAC)

In the OpenSSL libraries, the corresponding method object is `EVP_MAC`. The number for this operation is `OSSL_OP_MAC`. The functions the provider can offer are described in `provider-mac(7)`

Key Derivation Function (KDF)

In the OpenSSL libraries, the corresponding method object is `EVP_KDF`. The number for this operation is `OSSL_OP_KDF`. The functions the provider can offer are described in `provider-kdf(7)`

Key Exchange

In the OpenSSL libraries, the corresponding method object is `EVP_KEYEXCH`. The number for this operation is `OSSL_OP_KEYEXCH`. The functions the provider can offer are described in `provider-keyexch(7)`

Asymmetric Ciphers

In the OpenSSL libraries, the corresponding method object is `EVP_ASYM_CIPHER`. The number for this operation is `OSSL_OP_ASYM_CIPHER`. The functions the provider can offer are described in `provider-asym_cipher(7)`

Asymmetric Key Encapsulation

In the OpenSSL libraries, the corresponding method object is `EVP_KEM`. The number for this operation is `OSSL_OP_KEM`. The functions the provider can offer are described in

provider-kem(7)

Encoding

In the OpenSSL libraries, the corresponding method object is `OSSL_ENCODER`. The number for this operation is `OSSL_OP_ENCODER`. The functions the provider can offer are described in `provider-encoder(7)`

Algorithm naming

Algorithm names are case insensitive. Any particular algorithm can have multiple aliases associated with it. The canonical OpenSSL naming scheme follows this format:

`ALGNAME[VERSION?][SUBNAME[VERSION?]][SIZE?][MODE?]`

`VERSION` is only present if there are multiple versions of an algorithm (e.g. MD2, MD4, MD5). It may be omitted if there is only one version.

`SUBNAME` may be present where multiple algorithms are combined together, e.g. MD5-SHA1.

`SIZE` is only present if multiple versions of an algorithm exist with different sizes (e.g. AES-128-CBC, AES-256-CBC)

`MODE` is only present where applicable.

Other aliases may exist for example where standards bodies or common practice use alternative names or names that OpenSSL has used historically.

OPENSSL PROVIDERS

OpenSSL provides a number of its own providers. These are the default, base, fips, legacy and null providers. See `crypto(7)` for an overview of these providers.

SEE ALSO

`EVP_DigestInit_ex(3)`, `EVP_EncryptInit_ex(3)`, `OSSL_LIB_CTX(3)`,
`EVP_set_default_properties(3)`, `EVP_MD_fetch(3)`, `EVP_CIPHER_fetch(3)`, `EVP_KEYMGMT_fetch(3)`,
`openssl-core.h(7)`, `provider-base(7)`, `provider-digest(7)`, `provider-cipher(7)`,
`provider-keyexch(7)`

HISTORY

The concept of providers and everything surrounding them was introduced in OpenSSL 3.0.

COPYRIGHT

Copyright 2019-2021 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>.