Providers

Standard Providers

Providers are containers for algorithm implementations. Whenever a cryptographic algorithm is used via the high level APIs a provider is selected. It is that provider implementation that actually does the required work. There are five providers distributed with OpenSSL. In the future we expect third parties to distribute their own providers which can be added to OpenSSL dynamically. Documentation about writing providers is available on the provider(7) manual page.

The Default Provider

The default provider collects together all of the standard built-in OpenSSL algorithm implementations. If an application doesn't specify anything else explicitly (e.g. in the application or via config), then this is the provider that will be used. It is loaded automatically the first time that we try to get an algorithm from a provider if no other provider has been loaded yet. If another provider has already been loaded then it won't be loaded automatically. Therefore if you want to use it in conjunction with other providers then you must load it explicitly.

This is a "built-in" provider which means that it is compiled and linked into the libcrypto library and does not exist as a separate standalone module.

The Legacy Provider

The legacy provider is a collection of legacy algorithms that are either no longer in common use or considered insecure and strongly discouraged from use. However, some applications may need to use these algorithms for backwards compatibility reasons. This provider is **not** loaded by default. This may mean that some applications upgrading from earlier versions of OpenSSL may find that some algorithms are no longer available unless they load the legacy provider explicitly.

Algorithms in the legacy provider include MD2, MD4, MDC2, RMD160, CAST5, BF (Blowfish), IDEA, SEED, RC2, RC4, RC5 and DES (but not 3DES).

The FIPS Provider

The FIPS provider contains a sub-set of the algorithm implementations available from the default provider, consisting of algorithms conforming to FIPS standards. It is intended that this provider will be FIPS140-2 validated.

In some cases there may be minor behavioural differences between algorithm implementations in this provider compared to the equivalent algorithm in the default provider. This is typically in order to conform to FIPS standards.

The Base Provider

The base provider contains a small sub-set of non-cryptographic algorithms available in the default provider. For example, it contains algorithms to serialize and deserialize keys to files. If you do not load the default provider then you should always load this one instead (in particular, if you are using the FIPS provider).

The Null Provider

The null provider is "built-in" to libcrypto and contains no algorithm implementations. In order to guarantee that the default provider is not automatically loaded, the null provider can be loaded instead.

This can be useful if you are using non-default library contexts and want to ensure that the default library context is never used unintentionally.

Loading Providers

Providers to be loaded can be specified in the OpenSSL config file. See the config(5) manual page for information about how to configure providers via the config file, and how to automatically activate them.

The following is a minimal config file example to load and activate both the legacy and the default provider in the default library context.

```
openssl_conf = openssl_init

[openssl_init]
providers = provider_sect

[provider_sect]
default = default_sect
legacy = legacy_sect

[default_sect]
activate = 1

[legacy_sect]
activate = 1
```

It is also possible to load providers programmatically. For example you can load the legacy provider into the default library context as shown below. Note that once you have explicitly loaded a provider into the library context the default provider will no longer be automatically loaded. Therefore you will often also want to explicitly load the default provider, as is done here:

```
#include <stdio.h>
#include <stdlib.h>

#include <openssl/provider.h>

int main(void)
{
    OSSL_PROVIDER *legacy;
    OSSL_PROVIDER *deflt;

    /* Load Multiple providers into the default (NULL) library context */
    legacy = OSSL_PROVIDER_load(NULL, "legacy");
    if (legacy == NULL) {
        printf("Failed to load Legacy provider\n");
        exit(EXIT_FAILURE);
    }
    deflt = OSSL_PROVIDER_load(NULL, "default");
    if (deflt == NULL) {
        printf("Failed to load Default provider\n");
        OSSL_PROVIDER_unload(legacy);
        exit(EXIT_FAILURE);
    }

    /* Rest of application */

    OSSL_PROVIDER_unload(legacy);
    OSSL_PROVIDER_unload(deflt);
    exit(EXIT_SUCCESS);
}
```