NAME

provider-object - A specification for a provider-native object abstraction

SYNOPSIS

#include <openssl/core_object.h>
#include <openssl/core_names.h>

DESCRIPTION

The provider-native object abstraction is a set of **OSSL_PARAM**(3) keys and values that can be used to pass provider-native objects to OpenSSL library code or between different provider operation implementations with the help of OpenSSL library code.

The intention is that certain provider-native operations can pass any sort of object that belong with other operations, or with OpenSSL library code.

An object may be passed in the following manners:

1. *By value*

    This means that the *object data* is passed as an octet string or an UTF8 string, which can be handled in diverse ways by other provided implementations. The encoding of the object depends on the context it's used in; for example, **OSSL_DECODER**(3) allows multiple encodings, depending on existing decoders. If central OpenSSL library functionality is to handle the data directly, it **must** be encoded in DER for all object types except for **OSSL_OBJECT_NAME** (see "Parameter reference" below), where it's assumed to a plain UTF8 string.

2. *By reference*

    This means that the *object data* isn't passed directly, an *object reference* is passed instead. It's an octet string that only the correct provider understands correctly.

Objects *by value* can be used by anything that handles DER encoded objects.

Objects *by reference* need a higher level of cooperation from the implementation where the object originated (let's call it X) and its target implementation (let's call it Y):

1. *An object loading function in the target implementation*

    The target implementation (Y) may have a function that can take an *object reference*. This can only be used if the target implementation is from the same provider as the one originating the

object abstraction in question (X).

The exact target implementation to use is determined from the *object type* and possibly the *object data type*. For example, when the OpenSSL library receives an object abstraction with the *object type* **OSSL_OBJECT_PKEY**, it will fetch a **provider-keymgmt**(7) using the *object data type* as its key type (the second argument in **EVP_KEYMGMT_fetch**(3)).

2. *An object exporter in the originating implementation*

    The originating implementation (X) may have an exporter function. This exporter function can be used to export the object in **OSSL_PARAM**(3) form, that can then be imported by the target implementation's imported function.

    This can be used when it's not possible to fetch the target implementation (Y) from the same provider.

Parameter reference

A provider-native object abstraction is an **OSSL_PARAM**(3) with a selection of the following parameters:

"data" (**OSSL_OBJECT_PARAM_DATA**) <octet string> or <UTF8 string>
    The object data *passed by value*.

"reference" (**OSSL_OBJECT_PARAM_REFERENCE**) <octet string>
    The object data *passed by reference*.

"type" (**OSSL_OBJECT_PARAM_TYPE**) <integer>
    The *object type*, a number that may have any of the following values (all defined in *<openssl/core_object.h>*):

    OSSL_OBJECT_NAME
        The object data may only be *passed by value*, and should be a UTF8 string.

        This is useful for **provider-storemgmt**(7) when a URI load results in new URIs.

    OSSL_OBJECT_PKEY
        The object data is suitable as provider-native **EVP_PKEY** key data. The object data may be *passed by value* or *passed by reference*.

    OSSL_OBJECT_CERT

The object data is suitable as **X509** data. The object data for this object type can only be *passed by value*, and should be an octet string.

Since there's no provider-native X.509 object, OpenSSL libraries that receive this object abstraction are expected to convert the data to a **X509** object with **d2i_X509()**.

OSSL_OBJECT_CRL

The object data is suitable as **X509_CRL** data. The object data can only be *passed by value*, and should be an octet string.

Since there's no provider-native X.509 CRL object, OpenSSL libraries that receive this object abstraction are expected to convert the data to a **X509_CRL** object with **d2i_X509_CRL()**.

"data-type" (**OSSL_OBJECT_PARAM_DATA_TYPE**) <UTF8 string>
The specific type of the object content. Legitimate values depend on the object type; if it is **OSSL_OBJECT_PKEY**, the data type is expected to be a key type suitable for fetching a **provider-keymgmt**(7) that can handle the data.

"data-structure" (**OSSL_OBJECT_PARAM_DATA_STRUCTURE**) <UTF8 string>
The outermost structure of the object content. Legitimate values depend on the object type.

"desc" (**OSSL_OBJECT_PARAM_DESC**) <UTF8 string>
A human readable text that describes extra details on the object.

When a provider-native object abstraction is used, it *must* contain object data in at least one form (object data *passed by value*, i.e. the "data" item, or object data *passed by reference*, i.e. the "reference" item). Both may be present at once, in which case the OpenSSL library code that receives this will use the most optimal variant.

For objects with the object type **OSSL_OBJECT_NAME**, that object type *must* be given.

SEE ALSO

provider(7), **OSSL_DECODER**(3)

HISTORY

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

COPYRIGHT

Copyright 2020-2023 The OpenSSL Project Authors. All Rights Reserved.