

NAME

provider-kem - The kem library <-> provider functions

SYNOPSIS

```
#include <openssl/core_dispatch.h>
#include <openssl/core_names.h>

/*
 * None of these are actual functions, but are displayed like this for
 * the function signatures for functions that are offered as function
 * pointers in OSSL_DISPATCH arrays.
 */

/* Context management */
void *OSSL_FUNC_kem_newctx(void *provctx);
void OSSL_FUNC_kem_freectx(void *ctx);
void *OSSL_FUNC_kem_dupctx(void *ctx);

/* Encapsulation */
int OSSL_FUNC_kem_encapsulate_init(void *ctx, void *provkey, const char *name,
                                   const OSSL_PARAM params[]);
int OSSL_FUNC_kem_encapsulate(void *ctx, unsigned char *out, size_t *outlen,
                               unsigned char *secret, size_t *secretlen);

/* Decapsulation */
int OSSL_FUNC_kem_decapsulate_init(void *ctx, void *provkey, const char *name);
int OSSL_FUNC_kem_decapsulate(void *ctx, unsigned char *out, size_t *outlen,
                               const unsigned char *in, size_t inlen);

/* KEM parameters */
int OSSL_FUNC_kem_get_ctx_params(void *ctx, OSSL_PARAM params[]);
const OSSL_PARAM *OSSL_FUNC_kem_gettable_ctx_params(void *ctx, void *provctx);
int OSSL_FUNC_kem_set_ctx_params(void *ctx, const OSSL_PARAM params[]);
const OSSL_PARAM *OSSL_FUNC_kem_settable_ctx_params(void *ctx, void *provctx);
```

DESCRIPTION

This documentation is primarily aimed at provider authors. See **provider(7)** for further information.

The asymmetric kem (OSSL_OP_KEM) operation enables providers to implement asymmetric kem algorithms and make them available to applications via the API functions **EVP_PKEY_encapsulate(3)**,

EVP_PKEY_decapsulate(3) and other related functions.

All "functions" mentioned here are passed as function pointers between *libcrypto* and the provider in **OSSL_DISPATCH**(3) arrays via **OSSL_ALGORITHM**(3) arrays that are returned by the provider's **provider_query_operation**() function (see "Provider Functions" in **provider-base**(7)).

All these "functions" have a corresponding function type definition named **OSSL_FUNC_{name}_fn**, and a helper function to retrieve the function pointer from an **OSSL_DISPATCH**(3) element named **OSSL_FUNC_{name}**. For example, the "function" **OSSL_FUNC_kem_newctx**() has these:

```
typedef void *(OSSL_FUNC_kem_newctx_fn)(void *provctx);
static ossl_inline OSSL_FUNC_kem_newctx_fn
    OSSL_FUNC_kem_newctx(const OSSL_DISPATCH *opf);
```

OSSL_DISPATCH(3) arrays are indexed by numbers that are provided as macros in **openssl-core_dispatch.h**(7), as follows:

```
OSSL_FUNC_kem_newctx          OSSL_FUNC_KEM_NEWCTX
OSSL_FUNC_kem_freectx        OSSL_FUNC_KEM_FREECTX
OSSL_FUNC_kem_dupctx         OSSL_FUNC_KEM_DUPCTX

OSSL_FUNC_kem_encapsulate_init OSSL_FUNC_KEM_ENCAPSULATE_INIT
OSSL_FUNC_kem_encapsulate     OSSL_FUNC_KEM_ENCAPSULATE

OSSL_FUNC_kem_decapsulate_init OSSL_FUNC_KEM_DECAPSULATE_INIT
OSSL_FUNC_kem_decapsulate     OSSL_FUNC_KEM_DECAPSULATE

OSSL_FUNC_kem_get_ctx_params  OSSL_FUNC_KEM_GET_CTX_PARAMS
OSSL_FUNC_kem_gettable_ctx_params OSSL_FUNC_KEM_GETTABLE_CTX_PARAMS
OSSL_FUNC_kem_set_ctx_params  OSSL_FUNC_KEM_SET_CTX_PARAMS
OSSL_FUNC_kem_settable_ctx_params OSSL_FUNC_KEM_SETTABLE_CTX_PARAMS
```

An asymmetric kem algorithm implementation may not implement all of these functions. In order to be a consistent set of functions a provider must implement **OSSL_FUNC_kem_newctx** and **OSSL_FUNC_kem_freectx**. It must also implement both of **OSSL_FUNC_kem_encapsulate_init** and **OSSL_FUNC_kem_encapsulate**, or both of **OSSL_FUNC_kem_decapsulate_init** and **OSSL_FUNC_kem_decapsulate**. **OSSL_FUNC_kem_get_ctx_params** is optional but if it is present then so must **OSSL_FUNC_kem_gettable_ctx_params**. Similarly, **OSSL_FUNC_kem_set_ctx_params** is optional but if it is present then so must **OSSL_FUNC_kem_settable_ctx_params**.

An asymmetric kem algorithm must also implement some mechanism for generating, loading or importing keys via the key management (OSSL_OP_KEYMGMT) operation. See **provider-keymgmt(7)** for further details.

Context Management Functions

OSSL_FUNC_kem_newctx() should create and return a pointer to a provider side structure for holding context information during an asymmetric kem operation. A pointer to this context will be passed back in a number of the other asymmetric kem operation function calls. The parameter *provctx* is the provider context generated during provider initialisation (see **provider(7)**).

OSSL_FUNC_kem_freectx() is passed a pointer to the provider side asymmetric kem context in the *ctx* parameter. This function should free any resources associated with that context.

OSSL_FUNC_kem_dupctx() should duplicate the provider side asymmetric kem context in the *ctx* parameter and return the duplicate copy.

Asymmetric Key Encapsulation Functions

OSSL_FUNC_kem_encapsulate_init() initialises a context for an asymmetric encapsulation given a provider side asymmetric kem context in the *ctx* parameter, a pointer to a provider key object in the *provkey* parameter and the *name* of the algorithm. The *params*, if not NULL, should be set on the context in a manner similar to using **OSSL_FUNC_kem_set_ctx_params()**. The key object should have been previously generated, loaded or imported into the provider using the key management (OSSL_OP_KEYMGMT) operation (see **provider-keymgmt(7)**).

OSSL_FUNC_kem_encapsulate() performs the actual encapsulation itself. A previously initialised asymmetric kem context is passed in the *ctx* parameter. Unless *out* is NULL, the data to be encapsulated is internally generated, and returned into the buffer pointed to by the *secret* parameter and the encapsulated data should also be written to the location pointed to by the *out* parameter. The length of the encapsulated data should be written to **outlen* and the length of the generated secret should be written to **secretlen*.

If *out* is NULL then the maximum length of the encapsulated data should be written to **outlen*, and the maximum length of the generated secret should be written to **secretlen*.

Decapsulation Functions

OSSL_FUNC_kem_decapsulate_init() initialises a context for an asymmetric decapsulation given a provider side asymmetric kem context in the *ctx* parameter, a pointer to a provider key object in the *provkey* parameter, and a *name* of the algorithm. The key object should have been previously generated, loaded or imported into the provider using the key management (OSSL_OP_KEYMGMT) operation (see **provider-keymgmt(7)**).

OSSL_FUNC_kem_decapsulate() performs the actual decapsulation itself. A previously initialised asymmetric kem context is passed in the *ctx* parameter. The data to be decapsulated is pointed to by the *in* parameter which is *inlen* bytes long. Unless *out* is NULL, the decapsulated data should be written to the location pointed to by the *out* parameter. The length of the decapsulated data should be written to **outlen*. If *out* is NULL then the maximum length of the decapsulated data should be written to **outlen*.

Asymmetric Key Encapsulation Parameters

See **OSSL_PARAM(3)** for further details on the parameters structure used by the **OSSL_FUNC_kem_get_ctx_params()** and **OSSL_FUNC_kem_set_ctx_params()** functions.

OSSL_FUNC_kem_get_ctx_params() gets asymmetric kem parameters associated with the given provider side asymmetric kem context *ctx* and stores them in *params*. Passing NULL for *params* should return true.

OSSL_FUNC_kem_set_ctx_params() sets the asymmetric kem parameters associated with the given provider side asymmetric kem context *ctx* to *params*. Any parameter settings are additional to any that were previously set. Passing NULL for *params* should return true.

No parameters are currently recognised by built-in asymmetric kem algorithms.

OSSL_FUNC_kem_gettable_ctx_params() and **OSSL_FUNC_kem_settable_ctx_params()** get a constant **OSSL_PARAM(3)** array that describes the gettable and settable parameters, i.e. parameters that can be used with **OSSL_FUNC_kem_get_ctx_params()** and **OSSL_FUNC_kem_set_ctx_params()** respectively.

RETURN VALUES

OSSL_FUNC_kem_newctx() and **OSSL_FUNC_kem_dupctx()** should return the newly created provider side asymmetric kem context, or NULL on failure.

All other functions should return 1 for success or 0 on error.

SEE ALSO

provider(7)

HISTORY

The provider KEM interface was introduced in OpenSSL 3.0.

COPYRIGHT

Copyright 2020-2021 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>.