NAME

provider-kdf - The KDF 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_kdf_newctx(void *provctx);
void OSSL_FUNC_kdf_freectx(void *kctx);
void *OSSL_FUNC_kdf_dupctx(void *src);

/* Encryption/decryption */
int OSSL_FUNC_kdf_reset(void *kctx);
int OSSL_FUNC_kdf_derive(void *kctx, unsigned char *key, size_t keylen,
                         const OSSL_PARAM params[]);

/* KDF parameter descriptors */
const OSSL_PARAM *OSSL_FUNC_kdf_gettable_params(void *provctx);
const OSSL_PARAM *OSSL_FUNC_kdf_gettable_ctx_params(void *kcxt, void *provctx);
const OSSL_PARAM *OSSL_FUNC_kdf_settable_ctx_params(void *kcxt, void *provctx);

/* KDF parameters */
int OSSL_FUNC_kdf_get_params(OSSL_PARAM params[]);
int OSSL_FUNC_kdf_get_ctx_params(void *kctx, OSSL_PARAM params[]);
int OSSL_FUNC_kdf_set_ctx_params(void *kctx, const OSSL_PARAM params[]);
```

DESCRIPTION

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

The KDF operation enables providers to implement KDF algorithms and make them available to applications via the API functions **EVP_KDF_CTX_reset**(3), and **EVP_KDF_derive**(3).

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_kdf_newctx()** has these:

```
typedef void *(OSSL_FUNC_kdf_newctx_fn)(void *provctx);
static ossl_inline OSSL_FUNC_kdf_newctx_fn
    OSSL_FUNC_kdf_newctx(const OSSL_DISPATCH *opf);
```

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

```
OSSL_FUNC_kdf_newctx            OSSL_FUNC_KDF_NEWCTX
OSSL_FUNC_kdf_freectx           OSSL_FUNC_KDF_FREECTX
OSSL_FUNC_kdf_dupctx            OSSL_FUNC_KDF_DUPCTX

OSSL_FUNC_kdf_reset             OSSL_FUNC_KDF_RESET
OSSL_FUNC_kdf_derive            OSSL_FUNC_KDF_DERIVE

OSSL_FUNC_kdf_get_params        OSSL_FUNC_KDF_GET_PARAMS
OSSL_FUNC_kdf_get_ctx_params    OSSL_FUNC_KDF_GET_CTX_PARAMS
OSSL_FUNC_kdf_set_ctx_params    OSSL_FUNC_KDF_SET_CTX_PARAMS

OSSL_FUNC_kdf_gettable_params   OSSL_FUNC_KDF_GETTABLE_PARAMS
OSSL_FUNC_kdf_gettable_ctx_params  OSSL_FUNC_KDF_GETTABLE_CTX_PARAMS
OSSL_FUNC_kdf_settable_ctx_params  OSSL_FUNC_KDF_SETTABLE_CTX_PARAMS
```

A KDF algorithm implementation may not implement all of these functions. In order to be a consistent set of functions, at least the following functions must be implemented: **OSSL_FUNC_kdf_newctx()**, **OSSL_FUNC_kdf_freectx()**, **OSSL_FUNC_kdf_set_ctx_params()**, **OSSL_FUNC_kdf_derive()**. All other functions are optional.

Context Management Functions

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

OSSL_FUNC_kdf_freectx() is passed a pointer to the provider side KDF context in the *kctx* parameter. If it receives NULL as *kctx* value, it should not do anything other than return. This function should free any resources associated with that context.

OSSL_FUNC_kdf_dupctx() should duplicate the provider side KDF context in the *kctx* parameter and return the duplicate copy.

Encryption/Decryption Functions

OSSL_FUNC_kdf_reset() initialises a KDF operation given a provider side KDF context in the *kctx* parameter.

OSSL_FUNC_kdf_derive() performs the KDF operation after processing the *params* as per **OSSL_FUNC_kdf_set_ctx_params()**. The *kctx* parameter contains a pointer to the provider side context. The resulting key of the desired *keylen* should be written to *key*. If the algorithm does not support the requested *keylen* the function must return error.

KDF Parameters

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

OSSL_FUNC_kdf_get_params() gets details of parameter values associated with the provider algorithm and stores them in *params*.

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

OSSL_FUNC_kdf_get_ctx_params() retrieves gettable parameter values associated with the given provider side KDF context *kctx* and stores them in *params*. Passing NULL for *params* should return true.

OSSL_FUNC_kdf_gettable_params(), **OSSL_FUNC_kdf_gettable_ctx_params()**, and **OSSL_FUNC_kdf_settable_ctx_params()** all return constant **OSSL_PARAM**(3) arrays as descriptors of the parameters that **OSSL_FUNC_kdf_get_params()**, **OSSL_FUNC_kdf_get_ctx_params()**, and **OSSL_FUNC_kdf_set_ctx_params()** can handle, respectively.
OSSL_FUNC_kdf_gettable_ctx_params() and **OSSL_FUNC_kdf_settable_ctx_params()** will return the parameters associated with the provider side context *kctx* in its current state if it is not NULL. Otherwise, they return the parameters associated with the provider side algorithm *provctx*.

Parameters currently recognised by built-in KDFs are as follows. Not all parameters are relevant to, or are understood by all KDFs:

"size" (**OSSL_KDF_PARAM_SIZE**) <unsigned integer>
Gets the output size from the associated KDF ctx. If the algorithm produces a variable amount of output, SIZE_MAX should be returned. If the input parameters required to calculate the fixed output size have not yet been supplied, 0 should be returned indicating an error.

"key" (**OSSL_KDF_PARAM_KEY**) <octet string>
Sets the key in the associated KDF ctx.

"secret" (**OSSL_KDF_PARAM_SECRET**) <octet string>
Sets the secret in the associated KDF ctx.

"pass" (**OSSL_KDF_PARAM_PASSWORD**) <octet string>
Sets the password in the associated KDF ctx.

"cipher" (**OSSL_KDF_PARAM_CIPHER**) <UTF8 string>
"digest" (**OSSL_KDF_PARAM_DIGEST**) <UTF8 string>
"mac" (**OSSL_KDF_PARAM_MAC**) <UTF8 string>
Sets the name of the underlying cipher, digest or MAC to be used. It must name a suitable algorithm for the KDF that’s being used.

"maclen" (**OSSL_KDF_PARAM_MAC_SIZE**) <octet string>
Sets the length of the MAC in the associated KDF ctx.

"properties" (**OSSL_KDF_PARAM_PROPERTIES**) <UTF8 string>
Sets the properties to be queried when trying to fetch the underlying algorithm. This must be given together with the algorithm naming parameter to be considered valid.

"iter" (**OSSL_KDF_PARAM_ITER**) <unsigned integer>
Sets the number of iterations in the associated KDF ctx.

"mode" (**OSSL_KDF_PARAM_MODE**) <UTF8 string>
Sets the mode in the associated KDF ctx.

"pkcs5" (**OSSL_KDF_PARAM_PKCS5**) <integer>
Enables or disables the SP800-132 compliance checks. A mode of 0 enables the compliance checks.

The checks performed are:

- the iteration count is at least 1000.

- the salt length is at least 128 bits.
- the derived key length is at least 112 bits.

"ukm" (**OSSL_KDF_PARAM_UKM**) <octet string>
Sets an optional random string that is provided by the sender called "partyAInfo". In CMS this is the user keying material.

"cekalg" (**OSSL_KDF_PARAM_CEK_ALG**) <UTF8 string>
Sets the CEK wrapping algorithm name in the associated KDF ctx.

"n" (**OSSL_KDF_PARAM_SCRYPT_N**) <unsigned integer>
Sets the scrypt work factor parameter N in the associated KDF ctx.

"r" (**OSSL_KDF_PARAM_SCRYPT_R**) <unsigned integer>
Sets the scrypt work factor parameter r in the associated KDF ctx.

"p" (**OSSL_KDF_PARAM_SCRYPT_P**) <unsigned integer>
Sets the scrypt work factor parameter p in the associated KDF ctx.

"maxmem_bytes" (**OSSL_KDF_PARAM_SCRYPT_MAXMEM**) <unsigned integer>
Sets the scrypt work factor parameter maxmem in the associated KDF ctx.

"prefix" (**OSSL_KDF_PARAM_PREFIX**) <octet string>
Sets the prefix string using by the TLS 1.3 version of HKDF in the associated KDF ctx.

"label" (**OSSL_KDF_PARAM_LABEL**) <octet string>
Sets the label string using by the TLS 1.3 version of HKDF in the associated KDF ctx.

"data" (**OSSL_KDF_PARAM_DATA**) <octet string>
Sets the context string using by the TLS 1.3 version of HKDF in the associated KDF ctx.

"info" (**OSSL_KDF_PARAM_INFO**) <octet string>
Sets the optional shared info in the associated KDF ctx.

"seed" (**OSSL_KDF_PARAM_SEED**) <octet string>
Sets the IV in the associated KDF ctx.

"xcghash" (**OSSL_KDF_PARAM_SSHKDF_XCGHASH**) <octet string>
Sets the xcghash in the associated KDF ctx.

"session_id" (**OSSL_KDF_PARAM_SSHKDF_SESSION_ID**) <octet string>

Sets the session ID in the associated KDF ctx.

"type" (**OSSL_KDF_PARAM_SSHKDF_TYPE**) <UTF8 string>
Sets the SSH KDF type parameter in the associated KDF ctx. There are six supported types:

EVP_KDF_SSHKDF_TYPE_INITIAL_IV_CLI_TO_SRV
The Initial IV from client to server. A single char of value 65 (ASCII char 'A').

EVP_KDF_SSHKDF_TYPE_INITIAL_IV_SRV_TO_CLI
The Initial IV from server to client A single char of value 66 (ASCII char 'B').

EVP_KDF_SSHKDF_TYPE_ENCRYPTION_KEY_CLI_TO_SRV
The Encryption Key from client to server A single char of value 67 (ASCII char 'C').

EVP_KDF_SSHKDF_TYPE_ENCRYPTION_KEY_SRV_TO_CLI
The Encryption Key from server to client A single char of value 68 (ASCII char 'D').

EVP_KDF_SSHKDF_TYPE_INTEGRITY_KEY_CLI_TO_SRV
The Integrity Key from client to server A single char of value 69 (ASCII char 'E').

EVP_KDF_SSHKDF_TYPE_INTEGRITY_KEY_SRV_TO_CLI
The Integrity Key from client to server A single char of value 70 (ASCII char 'F').

"constant" (**OSSL_KDF_PARAM_CONSTANT**) <octet string>
Sets the constant value in the associated KDF ctx.

"id" (**OSSL_KDF_PARAM_PKCS12_ID**) <integer>
Sets the intended usage of the output bits in the associated KDF ctx. It is defined as per RFC 7292 section B.3.

RETURN VALUES

OSSL_FUNC_kdf_newctx() and **OSSL_FUNC_kdf_dupctx()** should return the newly created provider side KDF context, or NULL on failure.

OSSL_FUNC_kdf_derive(), **OSSL_FUNC_kdf_get_params()**, **OSSL_FUNC_kdf_get_ctx_params()** and **OSSL_FUNC_kdf_set_ctx_params()** should return 1 for success or 0 on error.

OSSL_FUNC_kdf_gettable_params(), **OSSL_FUNC_kdf_gettable_ctx_params()** and **OSSL_FUNC_kdf_settable_ctx_params()** should return a constant **OSSL_PARAM**(3) array, or NULL if none is offered.

NOTES

The KDF life-cycle is described in **life_cycle-kdf**(7). Providers should ensure that the various transitions listed there are supported. At some point the EVP layer will begin enforcing the listed transitions.

SEE ALSO

provider(7), **life_cycle-kdf**(7), **EVP_KDF**(3).

HISTORY

The provider KDF interface was introduced in OpenSSL 3.0.

COPYRIGHT