NAME

EVP_KDF−KRB5KDF − The RFC3961 Krb5 KDF EVP_KDF implementation

DESCRIPTION

Support for computing the **KRB5KDF** KDF through the **EVP_KDF** API.

The EVP_KDF−KRB5KDF algorithm implements the key derivation function defined in RFC 3961, section 5.1 and is used by Krb5 to derive session keys. Three inputs are required to perform key derivation: a cipher, (for example AES−128−CBC), the initial key, and a constant.

Identity

''KRB5KDF'' is the name for this implementation; it can be used with the **EVP_KDF_fetch()** function.

Supported parameters

The supported parameters are:

''properties'' (**OSSL_KDF_PARAM_PROPERTIES**) <UTF8 string>
''cipher'' (**OSSL_KDF_PARAM_CIPHER**) <UTF8 string>
''key'' (**OSSL_KDF_PARAM_KEY**) <octet string>
These parameters work as described in ''PARAMETERS'' in **EVP_KDF** (3).

''constant'' (**OSSL_KDF_PARAM_CONSTANT**) <octet string>
This parameter sets the constant value for the KDF. If a value is already set, the contents are replaced.

NOTES

A context for KRB5KDF can be obtained by calling:

```
EVP_KDF *kdf = EVP_KDF_fetch(NULL, "KRB5KDF", NULL);
EVP_KDF_CTX *kctx = EVP_KDF_CTX_new(kdf);
```

The output length of the KRB5KDF derivation is specified via the *keylen* parameter to the **EVP_KDF_derive** (3) function, and MUST match the key length for the chosen cipher or an error is returned. Moreover, the constant's length must not exceed the block size of the cipher. Since the KRB5KDF output length depends on the chosen cipher, calling **EVP_KDF_CTX_get_kdf_size** (3) to obtain the requisite length returns the correct length only after the cipher is set. Prior to that **EVP_MAX_KEY_LENGTH** is returned. The caller must allocate a buffer of the correct length for the chosen cipher, and pass that buffer to the **EVP_KDF_derive** (3) function along with that length.

EXAMPLES

This example derives a key using the AES−128−CBC cipher:

```
EVP_KDF *kdf;
EVP_KDF_CTX *kctx;
unsigned char key[16] = "01234...";
unsigned char constant[] = "I'm a constant";
unsigned char out[16];
size_t outlen = sizeof(out);
OSSL_PARAM params[4], *p = params;

kdf = EVP_KDF_fetch(NULL, "KRB5KDF", NULL);
kctx = EVP_KDF_CTX_new(kdf);
EVP_KDF_free(kdf);

*p++ = OSSL_PARAM_construct_utf8_string(OSSL_KDF_PARAM_CIPHER,
                                        SN_aes_128_cbc,
                                        strlen(SN_aes_128_cbc));
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_KEY,
                                         key, (size_t)16);
*p++ = OSSL_PARAM_construct_octet_string(OSSL_KDF_PARAM_CONSTANT,
                                         constant, strlen(constant));
*p = OSSL_PARAM_construct_end();
```

```
if (EVP_KDF_derive(kctx, out, outlen, params) <= 0)
    /* Error */

EVP_KDF_CTX_free(kctx);
```

CONFORMING TO

RFC 3961

SEE ALSO

EVP_KDF (3), **EVP_KDF_CTX_free** (3), **EVP_KDF_CTX_get_kdf_size** (3), **EVP_KDF_derive** (3), ''PARAMETERS'' in **EVP_KDF** (3)

HISTORY

This functionality was added to OpenSSL 3.0.

COPYRIGHT