Rocky Enterprise Linux 9.2 Manual Pages on command 'DSA_meth_dup.3ossl'

$ man DSA_meth_dup.3ossl

DSA_METH_NEW(3ossl) OpenSSL DSA_METH_NEW(3ossl)

NAME

DSA_meth_new, DSA_meth_free, DSA_meth_dup, DSA_meth_get0_name,

DSA_meth_set1_name, DSA_meth_get_flags, DSA_meth_set_flags,

DSA_meth_get0_app_data, DSA_meth_set0_app_data, DSA_meth_get_sign,

DSA_meth_set_sign, DSA_meth_get_sign_setup, DSA_meth_set_sign_setup,

DSA_meth_get_verify, DSA_meth_set_verify, DSA_meth_get_mod_exp,

DSA_meth_set_mod_exp, DSA_meth_get_bn_mod_exp, DSA_meth_set_bn_mod_exp,

DSA_meth_get_init, DSA_meth_set_init, DSA_meth_get_finish,

DSA_meth_set_finish, DSA_meth_get_paramgen, DSA_meth_set_paramgen,

DSA_meth_get_keygen, DSA_meth_set_keygen - Routines to build up DSA

methods

SYNOPSIS

#include <openssl/dsa.h>

The following functions have been deprecated since OpenSSL 3.0, and can

be hidden entirely by defining OPENSSL_API_COMPAT with a suitable
version value, see openssl_user_macros(7):

```
DSA_METHOD *DSA_meth_new(const char *name, int flags);

void DSA_meth_free(DSA_METHOD *dsam);

DSA_METHOD *DSA_meth_dup(const DSA_METHOD *meth);

const char *DSA_meth_get0_name(const DSA_METHOD *dsam);
int DSA_meth_set1_name(DSA_METHOD *dsam, const char *name);

int DSA_meth_get_flags(const DSA_METHOD *dsam);
int DSA_meth_set_flags(DSA_METHOD *dsam, int flags);

void *DSA_meth_get0_app_data(const DSA_METHOD *dsam);
int DSA_meth_set0_app_data(DSA_METHOD *dsam, void *app_data);

DSA_SIG *(*DSA_meth_get_sign(const DSA_METHOD *dsam))(const unsigned char *,
                                                      int, DSA *);
int DSA_meth_set_sign(DSA_METHOD *dsam, DSA_SIG *(*sign)(const unsigned char *,
                                                         int, DSA *));

int (*DSA_meth_get_sign_setup(const DSA_METHOD *dsam))(DSA *, BN_CTX *,$
                                                       BIGNUM **, BIGNUM **);
int DSA_meth_set_sign_setup(DSA_METHOD *dsam, int (*sign_setup)(DSA *, BN_CTX *,
                                                                BIGNUM **, BIGNUM **));

int (*DSA_meth_get_verify(const DSA_METHOD *dsam))(const unsigned char *,
                                                   int, DSA_SIG *, DSA *);
int DSA_meth_set_verify(DSA_METHOD *dsam, int (*verify)(const unsigned char *,
                                                        int, DSA_SIG *, DSA *));
```

```
int (*DSA_meth_get_mod_exp(const DSA_METHOD *dsam))(DSA *dsa, BIGNUM *rr, BIGNUM *a1,
                                                    BIGNUM *p1, BIGNUM *a2, BIGNUM *p2,
                                                    BIGNUM *m, BN_CTX *ctx,
                                                    BN_MONT_CTX *in_mont);
int DSA_meth_set_mod_exp(DSA_METHOD *dsam, int (*mod_exp)(DSA *dsa, BIGNUM *rr,
                                                          BIGNUM *a1, BIGNUM *p1,
                                                          BIGNUM *a2, BIGNUM *p2,
                                                          BIGNUM *m, BN_CTX *ctx,
                                                          BN_MONT_CTX *mont));

int (*DSA_meth_get_bn_mod_exp(const DSA_METHOD *dsam))(DSA *dsa, BIGNUM *r, BIGNUM *a,
                                                       const BIGNUM *p, const BIGNUM *m,
                                                       BN_CTX *ctx, BN_MONT_CTX *mont);
int DSA_meth_set_bn_mod_exp(DSA_METHOD *dsam, int (*bn_mod_exp)(DSA *dsa,
                                                                BIGNUM *r,
                                                                BIGNUM *a,
                                                                const BIGNUM *p,
                                                                const BIGNUM *m,
                                                                BN_CTX *ctx,
                                                                BN_MONT_CTX *mont));

int (*DSA_meth_get_init(const DSA_METHOD *dsam))(DSA *);
int DSA_meth_set_init(DSA_METHOD *dsam, int (*init)(DSA *));

int (*DSA_meth_get_finish(const DSA_METHOD *dsam))(DSA *);
int DSA_meth_set_finish(DSA_METHOD *dsam, int (*finish)(DSA *));

int (*DSA_meth_get_paramgen(const DSA_METHOD *dsam))(DSA *, int,
                                                     const unsigned char *,
                                                     int, int *, unsigned long *,
                                                     BN_GENCB *);
int DSA_meth_set_paramgen(DSA_METHOD *dsam,
                          int (*paramgen)(DSA *, int, const unsigned char *,
```

int, int *, unsigned long *, BN_GENCB *));

int (*DSA_meth_get_keygen(const DSA_METHOD *dsam))(DSA *);
int DSA_meth_set_keygen(DSA_METHOD *dsam, int (*keygen)(DSA *));

DESCRIPTION

All of the functions described on this page are deprecated.
Applications and extension implementations should instead use the
OSSL_PROVIDER APIs.

The DSA_METHOD type is a structure used for the provision of custom DSA implementations. It provides a set of functions used by OpenSSL for the implementation of the various DSA capabilities.

DSA_meth_new() creates a new DSA_METHOD structure. It should be given a unique name and a set of flags. The name should be a NULL terminated string, which will be duplicated and stored in the DSA_METHOD object. It is the callers responsibility to free the original string. The flags will be used during the construction of a new DSA object based on this DSA_METHOD. Any new DSA object will have those flags set by default.

DSA_meth_dup() creates a duplicate copy of the DSA_METHOD object passed as a parameter. This might be useful for creating a new DSA_METHOD based on an existing one, but with some differences.

DSA_meth_free() destroys a DSA_METHOD structure and frees up any memory associated with it.

DSA_meth_get0_name() will return a pointer to the name of this DSA_METHOD. This is a pointer to the internal name string and so should not be freed by the caller. DSA_meth_set1_name() sets the name of the DSA_METHOD to name. The string is duplicated and the copy is stored in the DSA_METHOD structure, so the caller remains responsible for freeing

the memory associated with the name.

DSA_meth_get_flags() returns the current value of the flags associated with this DSA_METHOD. DSA_meth_set_flags() provides the ability to set these flags.

The functions DSA_meth_get0_app_data() and DSA_meth_set0_app_data() provide the ability to associate implementation specific data with the DSA_METHOD. It is the application's responsibility to free this data before the DSA_METHOD is freed via a call to DSA_meth_free().

DSA_meth_get_sign() and DSA_meth_set_sign() get and set the function used for creating a DSA signature respectively. This function will be called in response to the application calling DSA_do_sign() (or DSA_sign()). The parameters for the function have the same meaning as for DSA_do_sign().

DSA_meth_get_sign_setup() and DSA_meth_set_sign_setup() get and set the function used for precalculating the DSA signature values k^-1 and r. This function will be called in response to the application calling DSA_sign_setup(). The parameters for the function have the same meaning as for DSA_sign_setup().

DSA_meth_get_verify() and DSA_meth_set_verify() get and set the function used for verifying a DSA signature respectively. This function will be called in response to the application calling DSA_do_verify() (or DSA_verify()). The parameters for the function have the same meaning as for DSA_do_verify().

DSA_meth_get_mod_exp() and DSA_meth_set_mod_exp() get and set the function used for computing the following value:

rr = a1^p1 * a2^p2 mod m

This function will be called by the default OpenSSL method during verification of a DSA signature. The result is stored in the rr parameter. This function may be NULL.

DSA_meth_get_bn_mod_exp() and DSA_meth_set_bn_mod_exp() get and set the function used for computing the following value:

r = a ^ p mod m

This function will be called by the default OpenSSL function for DSA_sign_setup(). The result is stored in the r parameter. This function may be NULL.

DSA_meth_get_init() and DSA_meth_set_init() get and set the function used for creating a new DSA instance respectively. This function will be called in response to the application calling DSA_new() (if the current default DSA_METHOD is this one) or DSA_new_method(). The DSA_new() and DSA_new_method() functions will allocate the memory for the new DSA object, and a pointer to this newly allocated structure will be passed as a parameter to the function. This function may be NULL.

DSA_meth_get_finish() and DSA_meth_set_finish() get and set the function used for destroying an instance of a DSA object respectively. This function will be called in response to the application calling DSA_free(). A pointer to the DSA to be destroyed is passed as a parameter. The destroy function should be used for DSA implementation specific clean up. The memory for the DSA itself should not be freed by this function. This function may be NULL.

DSA_meth_get_paramgen() and DSA_meth_set_paramgen() get and set the function used for generating DSA parameters respectively. This function

will be called in response to the application calling DSA_generate_parameters_ex() (or DSA_generate_parameters()). The parameters for the function have the same meaning as for DSA_generate_parameters_ex().

DSA_meth_get_keygen() and DSA_meth_set_keygen() get and set the function used for generating a new DSA key pair respectively. This function will be called in response to the application calling DSA_generate_key(). The parameter for the function has the same meaning as for DSA_generate_key().

RETURN VALUES

DSA_meth_new() and DSA_meth_dup() return the newly allocated DSA_METHOD object or NULL on failure.

DSA_meth_get0_name() and DSA_meth_get_flags() return the name and flags associated with the DSA_METHOD respectively.

All other DSA_meth_get_*() functions return the appropriate function pointer that has been set in the DSA_METHOD, or NULL if no such pointer has yet been set.

DSA_meth_set1_name() and all DSA_meth_set_*() functions return 1 on success or 0 on failure.

SEE ALSO

DSA_new(3), DSA_new(3), DSA_generate_parameters(3), DSA_generate_key(3), DSA_dup_DH(3), DSA_do_sign(3), DSA_set_method(3), DSA_SIG_new(3), DSA_sign(3), DSA_size(3), DSA_get0_pqg(3)

HISTORY

The functions described here were deprecated in OpenSSL 3.0.

The functions described here were added in OpenSSL 1.1.0.

COPYRIGHT