



Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'EVP_PKEY_meth_set_verifyctx.3ossl' command

`$ man EVP_PKEY_meth_set_verifyctx.3ossl`

EVP_PKEY_METH_NEW(3ossl) OpenSSL EVP_PKEY_METH_NEW(3ossl)

NAME

EVP_PKEY_meth_new, EVP_PKEY_meth_free, EVP_PKEY_meth_copy,
EVP_PKEY_meth_find, EVP_PKEY_meth_add0, EVP_PKEY_METHOD,
EVP_PKEY_meth_set_init, EVP_PKEY_meth_set_copy,
EVP_PKEY_meth_set_cleanup, EVP_PKEY_meth_set_paramgen,
EVP_PKEY_meth_set_keygen, EVP_PKEY_meth_set_sign,
EVP_PKEY_meth_set_verify, EVP_PKEY_meth_set_verify_recover,
EVP_PKEY_meth_set_signctx, EVP_PKEY_meth_set_verifyctx,
EVP_PKEY_meth_set_encrypt, EVP_PKEY_meth_set_decrypt,
EVP_PKEY_meth_set_derive, EVP_PKEY_meth_set_ctrl,
EVP_PKEY_meth_set_digestsign, EVP_PKEY_meth_set_digestverify,
EVP_PKEY_meth_set_check, EVP_PKEY_meth_set_public_check,
EVP_PKEY_meth_set_param_check, EVP_PKEY_meth_set_digest_custom,
EVP_PKEY_meth_get_init, EVP_PKEY_meth_get_copy,
EVP_PKEY_meth_get_cleanup, EVP_PKEY_meth_get_paramgen,
EVP_PKEY_meth_get_keygen, EVP_PKEY_meth_get_sign,
EVP_PKEY_meth_get_verify, EVP_PKEY_meth_get_verify_recover,
EVP_PKEY_meth_get_signctx, EVP_PKEY_meth_get_verifyctx,
EVP_PKEY_meth_get_encrypt, EVP_PKEY_meth_get_decrypt,
EVP_PKEY_meth_get_derive, EVP_PKEY_meth_get_ctrl,
EVP_PKEY_meth_get_digestsign, EVP_PKEY_meth_get_digestverify,
EVP_PKEY_meth_get_check, EVP_PKEY_meth_get_public_check,

EVP_PKEY_meth_get_param_check, EVP_PKEY_meth_get_digest_custom,
EVP_PKEY_meth_remove - manipulating EVP_PKEY_METHOD structure

SYNOPSIS

```
#include <openssl/evp.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):

```
typedef struct evp_pkey_method_st EVP_PKEY_METHOD;  
EVP_PKEY_METHOD *EVP_PKEY_meth_new(int id, int flags);  
void EVP_PKEY_meth_free(EVP_PKEY_METHOD *pmeth);  
void EVP_PKEY_meth_copy(EVP_PKEY_METHOD *dst, const EVP_PKEY_METHOD *src);  
const EVP_PKEY_METHOD *EVP_PKEY_meth_find(int type);  
int EVP_PKEY_meth_add0(const EVP_PKEY_METHOD *pmeth);  
int EVP_PKEY_meth_remove(const EVP_PKEY_METHOD *pmeth);  
void EVP_PKEY_meth_set_init(EVP_PKEY_METHOD *pmeth,  
                           int (*init) (EVP_PKEY_CTX *ctx));  
void EVP_PKEY_meth_set_copy(EVP_PKEY_METHOD *pmeth,  
                           int (*copy) (EVP_PKEY_CTX *dst,  
                                         const EVP_PKEY_CTX *src));  
void EVP_PKEY_meth_set_cleanup(EVP_PKEY_METHOD *pmeth,  
                              void (*cleanup) (EVP_PKEY_CTX *ctx));  
void EVP_PKEY_meth_set_paramgen(EVP_PKEY_METHOD *pmeth,  
                                int (*paramgen_init) (EVP_PKEY_CTX *ctx),  
                                int (*paramgen) (EVP_PKEY_CTX *ctx,  
                                                  EVP_PKEY *pkey));  
void EVP_PKEY_meth_set_keygen(EVP_PKEY_METHOD *pmeth,  
                              int (*keygen_init) (EVP_PKEY_CTX *ctx),  
                              int (*keygen) (EVP_PKEY_CTX *ctx,  
                                              EVP_PKEY *pkey));  
void EVP_PKEY_meth_set_sign(EVP_PKEY_METHOD *pmeth,  
                            int (*sign_init) (EVP_PKEY_CTX *ctx),  
                            int (*sign) (EVP_PKEY_CTX *ctx,  
                                         unsigned char *sig, size_t *siglen,
```

```

        const unsigned char *tbs,
        size_t tbslen));

void EVP_PKEY_meth_set_verify(EVP_PKEY_METHOD *pmeth,
    int (*verify_init) (EVP_PKEY_CTX *ctx),
    int (*verify) (EVP_PKEY_CTX *ctx,
        const unsigned char *sig,
        size_t siglen,
        const unsigned char *tbs,
        size_t tbslen));

void EVP_PKEY_meth_set_verify_recover(EVP_PKEY_METHOD *pmeth,
    int (*verify_recover_init) (EVP_PKEY_CTX
        *ctx),
    int (*verify_recover) (EVP_PKEY_CTX
        *ctx,
        unsigned char
        *sig,
        size_t *siglen,
        const unsigned
        char *tbs,
        size_t tbslen));

void EVP_PKEY_meth_set_signctx(EVP_PKEY_METHOD *pmeth,
    int (*signctx_init) (EVP_PKEY_CTX *ctx,
        EVP_MD_CTX *mctx),
    int (*signctx) (EVP_PKEY_CTX *ctx,
        unsigned char *sig,
        size_t *siglen,
        EVP_MD_CTX *mctx));

void EVP_PKEY_meth_set_verifyctx(EVP_PKEY_METHOD *pmeth,
    int (*verifyctx_init) (EVP_PKEY_CTX *ctx,
        EVP_MD_CTX *mctx),
    int (*verifyctx) (EVP_PKEY_CTX *ctx,
        const unsigned char *sig,
        int siglen,

```

```

        EVP_MD_CTX *mctx));

void EVP_PKEY_meth_set_encrypt(EVP_PKEY_METHOD *pmeth,
    int (*encrypt_init) (EVP_PKEY_CTX *ctx),
    int (*encryptfn) (EVP_PKEY_CTX *ctx,
        unsigned char *out,
        size_t *outlen,
        const unsigned char *in,
        size_t inlen));

void EVP_PKEY_meth_set_decrypt(EVP_PKEY_METHOD *pmeth,
    int (*decrypt_init) (EVP_PKEY_CTX *ctx),
    int (*decrypt) (EVP_PKEY_CTX *ctx,
        unsigned char *out,
        size_t *outlen,
        const unsigned char *in,
        size_t inlen));

void EVP_PKEY_meth_set_derive(EVP_PKEY_METHOD *pmeth,
    int (*derive_init) (EVP_PKEY_CTX *ctx),
    int (*derive) (EVP_PKEY_CTX *ctx,
        unsigned char *key,
        size_t *keylen));

void EVP_PKEY_meth_set_ctrl(EVP_PKEY_METHOD *pmeth,
    int (*ctrl) (EVP_PKEY_CTX *ctx, int type, int p1,
        void *p2),
    int (*ctrl_str) (EVP_PKEY_CTX *ctx,
        const char *type,
        const char *value));

void EVP_PKEY_meth_set_digestsign(EVP_PKEY_METHOD *pmeth,
    int (*digestsign) (EVP_MD_CTX *ctx,
        unsigned char *sig,
        size_t *siglen,
        const unsigned char *tbs,
        size_t tbslen));

void EVP_PKEY_meth_set_digestverify(EVP_PKEY_METHOD *pmeth,

```

```

        int (*digestverify) (EVP_MD_CTX *ctx,
                            const unsigned char *sig,
                            size_t siglen,
                            const unsigned char *tbs,
                            size_t tbslen));

void EVP_PKEY_meth_set_check(EVP_PKEY_METHOD *pmeth,
                            int (*check) (EVP_PKEY *pkey));

void EVP_PKEY_meth_set_public_check(EVP_PKEY_METHOD *pmeth,
                                    int (*check) (EVP_PKEY *pkey));

void EVP_PKEY_meth_set_param_check(EVP_PKEY_METHOD *pmeth,
                                   int (*check) (EVP_PKEY *pkey));

void EVP_PKEY_meth_set_digest_custom(EVP_PKEY_METHOD *pmeth,
                                     int (*digest_custom) (EVP_PKEY_CTX *ctx,
                                                           EVP_MD_CTX *mctx));

void EVP_PKEY_meth_get_init(const EVP_PKEY_METHOD *pmeth,
                           int (**pinit) (EVP_PKEY_CTX *ctx));

void EVP_PKEY_meth_get_copy(const EVP_PKEY_METHOD *pmeth,
                            int (**pcopy) (EVP_PKEY_CTX *dst,
                                             EVP_PKEY_CTX *src));

void EVP_PKEY_meth_get_cleanup(const EVP_PKEY_METHOD *pmeth,
                               void (**pcleanup) (EVP_PKEY_CTX *ctx));

void EVP_PKEY_meth_get_paramgen(const EVP_PKEY_METHOD *pmeth,
                                int (**pparamgen_init) (EVP_PKEY_CTX *ctx),
                                int (**pparamgen) (EVP_PKEY_CTX *ctx,
                                                    EVP_PKEY *pkey));

void EVP_PKEY_meth_get_keygen(const EVP_PKEY_METHOD *pmeth,
                              int (**pkeygen_init) (EVP_PKEY_CTX *ctx),
                              int (**pkeygen) (EVP_PKEY_CTX *ctx,
                                                EVP_PKEY *pkey));

void EVP_PKEY_meth_get_sign(const EVP_PKEY_METHOD *pmeth,
                            int (**psign_init) (EVP_PKEY_CTX *ctx),
                            int (**psign) (EVP_PKEY_CTX *ctx,
                                             unsigned char *sig, size_t *siglen,

```

```

        const unsigned char *tbs,
        size_t tbslen));

void EVP_PKEY_meth_get_verify(const EVP_PKEY_METHOD *pmeth,
    int (**pverify_init) (EVP_PKEY_CTX *ctx),
    int (**pverify) (EVP_PKEY_CTX *ctx,
        const unsigned char *sig,
        size_t siglen,
        const unsigned char *tbs,
        size_t tbslen));

void EVP_PKEY_meth_get_verify_recover(const EVP_PKEY_METHOD *pmeth,
    int (**pverify_recover_init) (EVP_PKEY_CTX
        *ctx),
    int (**pverify_recover) (EVP_PKEY_CTX
        *ctx,
        unsigned char
        *sig,
        size_t *siglen,
        const unsigned
        char *tbs,
        size_t tbslen));

void EVP_PKEY_meth_get_signctx(const EVP_PKEY_METHOD *pmeth,
    int (**psignctx_init) (EVP_PKEY_CTX *ctx,
        EVP_MD_CTX *mctx),
    int (**psignctx) (EVP_PKEY_CTX *ctx,
        unsigned char *sig,
        size_t *siglen,
        EVP_MD_CTX *mctx));

void EVP_PKEY_meth_get_verifyctx(const EVP_PKEY_METHOD *pmeth,
    int (**pverifyctx_init) (EVP_PKEY_CTX *ctx,
        EVP_MD_CTX *mctx),
    int (**pverifyctx) (EVP_PKEY_CTX *ctx,
        const unsigned char *sig,
        int siglen,

```

```

        EVP_MD_CTX *mctx));

void EVP_PKEY_meth_get_encrypt(const EVP_PKEY_METHOD *pmeth,
    int (**pencrypt_init) (EVP_PKEY_CTX *ctx),
    int (**pencryptfn) (EVP_PKEY_CTX *ctx,
        unsigned char *out,
        size_t *outlen,
        const unsigned char *in,
        size_t inlen));

void EVP_PKEY_meth_get_decrypt(const EVP_PKEY_METHOD *pmeth,
    int (**pdecrypt_init) (EVP_PKEY_CTX *ctx),
    int (**pdecrypt) (EVP_PKEY_CTX *ctx,
        unsigned char *out,
        size_t *outlen,
        const unsigned char *in,
        size_t inlen));

void EVP_PKEY_meth_get_derive(const EVP_PKEY_METHOD *pmeth,
    int (**pderive_init) (EVP_PKEY_CTX *ctx),
    int (**pderive) (EVP_PKEY_CTX *ctx,
        unsigned char *key,
        size_t *keylen));

void EVP_PKEY_meth_get_ctrl(const EVP_PKEY_METHOD *pmeth,
    int (**pctrl) (EVP_PKEY_CTX *ctx, int type, int p1,
        void *p2),
    int (**pctrl_str) (EVP_PKEY_CTX *ctx,
        const char *type,
        const char *value));

void EVP_PKEY_meth_get_digestsign(const EVP_PKEY_METHOD *pmeth,
    int (**digestsign) (EVP_MD_CTX *ctx,
        unsigned char *sig,
        size_t *siglen,
        const unsigned char *tbs,
        size_t tbslen));

void EVP_PKEY_meth_get_digestverify(const EVP_PKEY_METHOD *pmeth,

```

```

        int (**digestverify) (EVP_MD_CTX *ctx,
                               const unsigned char *sig,
                               size_t siglen,
                               const unsigned char *tbs,
                               size_t tbslen));

void EVP_PKEY_meth_get_check(const EVP_PKEY_METHOD *pmeth,
                             int (**pcheck) (EVP_PKEY *pkey));

void EVP_PKEY_meth_get_public_check(const EVP_PKEY_METHOD *pmeth,
                                     int (**pcheck) (EVP_PKEY *pkey));

void EVP_PKEY_meth_get_param_check(const EVP_PKEY_METHOD *pmeth,
                                    int (**pcheck) (EVP_PKEY *pkey));

void EVP_PKEY_meth_get_digest_custom(const EVP_PKEY_METHOD *pmeth,
                                     int (**pdigest_custom) (EVP_PKEY_CTX *ctx,
                                                             EVP_MD_CTX *mctx));

```

DESCRIPTION

All of the functions described on this page are deprecated.

Applications should instead use the OSSL_PROVIDER APIs.

EVP_PKEY_METHOD is a structure which holds a set of methods for a specific public key cryptographic algorithm. Those methods are usually used to perform different jobs, such as generating a key, signing or verifying, encrypting or decrypting, etc.

There are two places where the EVP_PKEY_METHOD objects are stored: one is a built-in static array representing the standard methods for different algorithms, and the other one is a stack of user-defined application-specific methods, which can be manipulated by using EVP_PKEY_meth_add0(3).

The EVP_PKEY_METHOD objects are usually referenced by EVP_PKEY_CTX objects.

Methods

The methods are the underlying implementations of a particular public key algorithm present by the EVP_PKEY_CTX object.

```

int (*init) (EVP_PKEY_CTX *ctx);

int (*copy) (EVP_PKEY_CTX *dst, const EVP_PKEY_CTX *src);

```

```
void (*cleanup) (EVP_PKEY_CTX *ctx);
```

The `init()` method is called to initialize algorithm-specific data when a new `EVP_PKEY_CTX` is created. As opposed to `init()`, the `cleanup()` method is called when an `EVP_PKEY_CTX` is freed. The `copy()` method is called when an `EVP_PKEY_CTX` is being duplicated. Refer to `EVP_PKEY_CTX_new(3)`, `EVP_PKEY_CTX_new_id(3)`, `EVP_PKEY_CTX_free(3)` and `EVP_PKEY_CTX_dup(3)`.

```
int (*paramgen_init) (EVP_PKEY_CTX *ctx);
```

```
int (*paramgen) (EVP_PKEY_CTX *ctx, EVP_PKEY *pkey);
```

The `paramgen_init()` and `paramgen()` methods deal with key parameter generation. They are called by `EVP_PKEY_paramgen_init(3)` and `EVP_PKEY_paramgen(3)` to handle the parameter generation process.

```
int (*keygen_init) (EVP_PKEY_CTX *ctx);
```

```
int (*keygen) (EVP_PKEY_CTX *ctx, EVP_PKEY *pkey);
```

The `keygen_init()` and `keygen()` methods are used to generate the actual key for the specified algorithm. They are called by `EVP_PKEY_keygen_init(3)` and `EVP_PKEY_keygen(3)`.

```
int (*sign_init) (EVP_PKEY_CTX *ctx);
```

```
int (*sign) (EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,  
            const unsigned char *tbs, size_t tbslen);
```

The `sign_init()` and `sign()` methods are used to generate the signature of a piece of data using a private key. They are called by `EVP_PKEY_sign_init(3)` and `EVP_PKEY_sign(3)`.

```
int (*verify_init) (EVP_PKEY_CTX *ctx);
```

```
int (*verify) (EVP_PKEY_CTX *ctx,  
             const unsigned char *sig, size_t siglen,  
             const unsigned char *tbs, size_t tbslen);
```

The `verify_init()` and `verify()` methods are used to verify whether a signature is valid. They are called by `EVP_PKEY_verify_init(3)` and `EVP_PKEY_verify(3)`.

```
int (*verify_recover_init) (EVP_PKEY_CTX *ctx);
```

```
int (*verify_recover) (EVP_PKEY_CTX *ctx,  
                    unsigned char *rout, size_t *routlen,
```

```
const unsigned char *sig, size_t siglen);
```

The `verify_recover_init()` and `verify_recover()` methods are used to verify a signature and then recover the digest from the signature (for instance, a signature that was generated by RSA signing algorithm).

They are called by `EVP_PKEY_verify_recover_init(3)` and `EVP_PKEY_verify_recover(3)`.

```
int (*signctx_init) (EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx);
```

```
int (*signctx) (EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,  
               EVP_MD_CTX *mctx);
```

The `signctx_init()` and `signctx()` methods are used to sign a digest present by a `EVP_MD_CTX` object. They are called by the `EVP_DigestSign` functions. See `EVP_DigestSignInit(3)` for details.

```
int (*verifyctx_init) (EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx);
```

```
int (*verifyctx) (EVP_PKEY_CTX *ctx, const unsigned char *sig, int siglen,  
                 EVP_MD_CTX *mctx);
```

The `verifyctx_init()` and `verifyctx()` methods are used to verify a signature against the data in a `EVP_MD_CTX` object. They are called by the various `EVP_DigestVerify` functions. See `EVP_DigestVerifyInit(3)` for details.

```
int (*encrypt_init) (EVP_PKEY_CTX *ctx);
```

```
int (*encrypt) (EVP_PKEY_CTX *ctx, unsigned char *out, size_t *outlen,  
               const unsigned char *in, size_t inlen);
```

The `encrypt_init()` and `encrypt()` methods are used to encrypt a piece of data. They are called by `EVP_PKEY_encrypt_init(3)` and `EVP_PKEY_encrypt(3)`.

```
int (*decrypt_init) (EVP_PKEY_CTX *ctx);
```

```
int (*decrypt) (EVP_PKEY_CTX *ctx, unsigned char *out, size_t *outlen,  
               const unsigned char *in, size_t inlen);
```

The `decrypt_init()` and `decrypt()` methods are used to decrypt a piece of data. They are called by `EVP_PKEY_decrypt_init(3)` and `EVP_PKEY_decrypt(3)`.

```
int (*derive_init) (EVP_PKEY_CTX *ctx);
```

```
int (*derive) (EVP_PKEY_CTX *ctx, unsigned char *key, size_t *keylen);
```

The `derive_init()` and `derive()` methods are used to derive the shared secret from a public key algorithm (for instance, the DH algorithm).

They are called by `EVP_PKEY_derive_init(3)` and `EVP_PKEY_derive(3)`.

```
int (*ctrl) (EVP_PKEY_CTX *ctx, int type, int p1, void *p2);
```

```
int (*ctrl_str) (EVP_PKEY_CTX *ctx, const char *type, const char *value);
```

The `ctrl()` and `ctrl_str()` methods are used to adjust algorithm-specific settings. See `EVP_PKEY_CTX_ctrl(3)` and related functions for details.

```
int (*digestsign) (EVP_MD_CTX *ctx, unsigned char *sig, size_t *siglen,  
                  const unsigned char *tbs, size_t tbslen);
```

```
int (*digestverify) (EVP_MD_CTX *ctx, const unsigned char *sig,  
                    size_t siglen, const unsigned char *tbs,  
                    size_t tbslen);
```

The `digestsign()` and `digestverify()` methods are used to generate or verify a signature in a one-shot mode. They could be called by `EVP_DigestSign(3)` and `EVP_DigestVerify(3)`.

```
int (*check) (EVP_PKEY *pkey);
```

```
int (*public_check) (EVP_PKEY *pkey);
```

```
int (*param_check) (EVP_PKEY *pkey);
```

The `check()`, `public_check()` and `param_check()` methods are used to validate a key-pair, the public component and parameters respectively for a given pkey. They could be called by `EVP_PKEY_check(3)`, `EVP_PKEY_public_check(3)` and `EVP_PKEY_param_check(3)` respectively.

```
int (*digest_custom) (EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx);
```

The `digest_custom()` method is used to generate customized digest content before the real message is passed to functions like `EVP_DigestSignUpdate(3)` or `EVP_DigestVerifyInit(3)`. This is usually required by some public key signature algorithms like SM2 which requires a hashed prefix to the message to be signed. The `digest_custom()` function will be called by `EVP_DigestSignInit(3)` and `EVP_DigestVerifyInit(3)`.

Functions

`EVP_PKEY_meth_new()` creates and returns a new `EVP_PKEY_METHOD` object, and associates the given id and flags. The following flags are

supported:

`EVP_PKEY_FLAG_AUTOARGLEN`

`EVP_PKEY_FLAG_SIGCTX_CUSTOM`

If an `EVP_PKEY_METHOD` is set with the `EVP_PKEY_FLAG_AUTOARGLEN` flag, the maximum size of the output buffer will be automatically calculated or checked in corresponding EVP methods by the EVP framework. Thus the implementations of these methods don't need to care about handling the case of returning output buffer size by themselves. For details on the output buffer size, refer to `EVP_PKEY_sign(3)`.

The `EVP_PKEY_FLAG_SIGCTX_CUSTOM` is used to indicate the `signctx()` method of an `EVP_PKEY_METHOD` is always called by the EVP framework while doing a digest signing operation by calling `EVP_DigestSignFinal(3)`.

`EVP_PKEY_meth_free()` frees an existing `EVP_PKEY_METHOD` pointed by `pmeth`.

`EVP_PKEY_meth_copy()` copies an `EVP_PKEY_METHOD` object from `src` to `dst`.

`EVP_PKEY_meth_find()` finds an `EVP_PKEY_METHOD` object with the `id`. This function first searches through the user-defined method objects and then the built-in objects.

`EVP_PKEY_meth_add0()` adds `pmeth` to the user defined stack of methods.

`EVP_PKEY_meth_remove()` removes an `EVP_PKEY_METHOD` object added by `EVP_PKEY_meth_add0()`.

The `EVP_PKEY_meth_set` functions set the corresponding fields of `EVP_PKEY_METHOD` structure with the arguments passed.

The `EVP_PKEY_meth_get` functions get the corresponding fields of `EVP_PKEY_METHOD` structure to the arguments provided.

RETURN VALUES

`EVP_PKEY_meth_new()` returns a pointer to a new `EVP_PKEY_METHOD` object or returns `NULL` on error.

`EVP_PKEY_meth_free()` and `EVP_PKEY_meth_copy()` do not return values.

`EVP_PKEY_meth_find()` returns a pointer to the found `EVP_PKEY_METHOD` object or returns `NULL` if not found.

`EVP_PKEY_meth_add0()` returns 1 if method is added successfully or 0 if

an error occurred.

EVP_PKEY_meth_remove() returns 1 if method is removed successfully or 0 if an error occurred.

All EVP_PKEY_meth_set and EVP_PKEY_meth_get functions have no return values. For the 'get' functions, function pointers are returned by arguments.

HISTORY

All of these functions were deprecated in OpenSSL 3.0.

The signature of the copy functional argument of EVP_PKEY_meth_set_copy() has changed in OpenSSL 3.0 so its src parameter is now constified.

COPYRIGHT

Copyright 2017-2021 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>.

3.0.7 2023-07-13 EVP_PKEY_METH_NEW(3ossl)