NAME

EVP_PKEY_CTX_ctrl, EVP_PKEY_CTX_ctrl_str, EVP_PKEY_get_default_digest_nid, EVP_PKEY_CTX_set_signature_md, EVP_PKEY_CTX_set_rsa_padding, EVP_PKEY_CTX_set_rsa_pss_saltlen, EVP_PKEY_CTX_set_rsa_rsa_keygen_bits, EVP_PKEY_CTX_set_rsa_keygen_pubexp, EVP_PKEY_CTX_set_dsa_paramgen_bits, EVP_PKEY_CTX_set_dh_paramgen_prime_len, EVP_PKEY_CTX_set_dh_paramgen_generator, EVP_PKEY_CTX_set_ec_paramgen_curve_nid - algorithm specific control operations

SYNOPSIS

```
#include <openssl/evp.h>

int EVP_PKEY_CTX_ctrl(EVP_PKEY_CTX *ctx, int keytype, int optype,
int cmd, int p1, void *p2);
int EVP_PKEY_CTX_ctrl_str(EVP_PKEY_CTX *ctx, const char *type,
const char *value);

int EVP_PKEY_get_default_digest_nid(EVP_PKEY *pkey, int *pnid);

#include <openssl/rsa.h>

int EVP_PKEY_CTX_set_signature_md(EVP_PKEY_CTX *ctx, const EVP_MD *md);

int EVP_PKEY_CTX_set_rsa_padding(EVP_PKEY_CTX *ctx, int pad);
int EVP_PKEY_CTX_set_rsa_pss_saltlen(EVP_PKEY_CTX *ctx, int len);
int EVP_PKEY_CTX_set_rsa_rsa_keygen_bits(EVP_PKEY_CTX *ctx, int mbits);
int EVP_PKEY_CTX_set_rsa_keygen_pubexp(EVP_PKEY_CTX *ctx, BIGNUM *pubexp);

#include <openssl/dsa.h>
int EVP_PKEY_CTX_set_dsa_paramgen_bits(EVP_PKEY_CTX *ctx, int nbits);

#include <openssl/dh.h>
int EVP_PKEY_CTX_set_dh_paramgen_prime_len(EVP_PKEY_CTX *ctx, int len);
int EVP_PKEY_CTX_set_dh_paramgen_generator(EVP_PKEY_CTX *ctx, int gen);

#include <openssl/ec.h>
int EVP_PKEY_CTX_set_ec_paramgen_curve_nid(EVP_PKEY_CTX *ctx, int nid);
```

DESCRIPTION

The function *EVP_PKEY_CTX_ctrl()* sends a control operation to the context **ctx**. The key type used must match **keytype** if it is not -1. The parameter **optype** is a mask indicating which operations the control can be applied to. The control command is indicated in **cmd** and any additional arguments in **p1** and **p2**.

Applications will not normally call *EVP_PKEY_CTX_ctrl()* directly but will instead call one of the algorithm specific macros below.

The function *EVP_PKEY_CTX_ctrl_str()* allows an application to send an algorithm specific control operation to a context **ctx** in string form. This is intended to be used for options specified on the command line or in text files. The commands supported are documented in the openssl utility command line pages for the option **-pkeyopt** which is supported by the **pkeyutl**, **genpkey** and **req** commands.

All the remaining ''functions'' are implemented as macros.

The *EVP_PKEY_CTX_set_signature_md()* macro sets the message digest type used in a signature. It can be used with any public key algorithm supporting signature operations.

The macro *EVP_PKEY_CTX_set_rsa_padding()* sets the RSA padding mode for **ctx**. The**pad** parameter can take the value RSA_PKCS1_PADDING for PKCS#1 padding,

RSA_SSLV23_PADDING for SSLv23 padding, RSA_NO_PADDING for no padding, RSA_PKCS1_OAEP_PADDING for OAEP padding (encrypt and decrypt only), RSA_X931_PADDING for X9.31 padding (signature operations only) and RSA_PKCS1_PSS_PADDING (sign and verify only).

Two RSA padding modes behave differently if *EVP_PKEY_CTX_set_signature_md()* is used. If this macro is called for PKCS#1 padding the plaintext buffer is an actual digest value and is encapsulated in a DigestInfo structure according to PKCS#1 when signing and this structure is expected (and stripped off) when verifying. If this control is not used with RSA and PKCS#1 padding then the supplied data is used directly and not encapsulated. In the case of X9.31 padding for RSA the algorithm identifier byte is added or checked and removed if this control is called. If it is not called then the first byte of the plaintext buffer is expected to be the algorithm identifier byte.

The *EVP_PKEY_CTX_set_rsa_pss_saltlen()* macro sets the RSA PSS salt length to **len** as its name implies it is only supported for PSS padding. Two special values are supported: -1 sets the salt length to the digest length. When signing -2 sets the salt length to the maximum permissible value. When verifying -2 causes the salt length to be automatically determined based on the **PSS** block structure. If this macro is not called a salt length value of -2 is used by default.

The *EVP_PKEY_CTX_set_rsa_rsa_keygen_bits()* macro sets the RSA key length for RSA key genration to **bits**. If not specified 1024 bits is used.

The *EVP_PKEY_CTX_set_rsa_keygen_pubexp()* macro sets the public exponent value for RSA key generation to **pubexp** currently it should be an odd integer. The **pubexp** pointer is used internally by this function so it should not be modified or free after the call. If this macro is not called then 65537 is used.

The macro *EVP_PKEY_CTX_set_dsa_paramgen_bits()* sets the number of bits used for DSA parameter generation to **bits**. If not specified 1024 is used.

The macro *EVP_PKEY_CTX_set_dh_paramgen_prime_len()* sets the length of the DH prime parameter **p** for DH parameter generation. If this macro is not called then 1024 is used.

The *EVP_PKEY_CTX_set_dh_paramgen_generator()* macro sets DH generator to **gen** for DH parameter generation. If not specified 2 is used.

The *EVP_PKEY_CTX_set_ec_paramgen_curve_nid()* sets the EC curve for EC parameter generation to **nid**. For EC parameter generation this macro must be called or an error occurs because there is no default curve.

RETURN VALUES

EVP_PKEY_CTX_ctrl() and its macros return a positive value for success and 0 or a negative value for failure. In particular a return value of -2 indicates the operation is not supported by the public key algorithm.

SEE ALSO

EVP_PKEY_CTX_new(3), *EVP_PKEY_encrypt(3)*, *EVP_PKEY_decrypt(3)*, *EVP_PKEY_sign(3)*, *EVP_PKEY_verify(3)*, *EVP_PKEY_verify_recover(3)*, *EVP_PKEY_derive(3)* *EVP_PKEY_keygen(3)*

HISTORY

These functions were first added to OpenSSL 1.0.0.