NAME

OSSL_CRMF_MSG_set0_validity, OSSL_CRMF_MSG_set_certReqId, OSSL_CRMF_CERTTEMPLATE_fill, OSSL_CRMF_MSG_set0_extensions, OSSL_CRMF_MSG_push0_extension, OSSL_CRMF_MSG_create_popo, OSSL_CRMF_MSGS_verify_popo - functions populating and verifying CRMF CertReqMsg structures

SYNOPSIS

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

int OSSL_CRMF_MSG_set0_validity(OSSL_CRMF_MSG *crm,
                                ASN1_TIME *notBefore, ASN1_TIME *notAfter);

int OSSL_CRMF_MSG_set_certReqId(OSSL_CRMF_MSG *crm, int rid);

int OSSL_CRMF_CERTTEMPLATE_fill(OSSL_CRMF_CERTTEMPLATE *tmpl,
                                EVP_PKEY *pubkey,
                                const X509_NAME *subject,
                                const X509_NAME *issuer,
                                const ASN1_INTEGER *serial);

int OSSL_CRMF_MSG_set0_extensions(OSSL_CRMF_MSG *crm, X509_EXTENSIONS *exts);

int OSSL_CRMF_MSG_push0_extension(OSSL_CRMF_MSG *crm, X509_EXTENSION *ext);

int OSSL_CRMF_MSG_create_popo(int meth, OSSL_CRMF_MSG *crm,
                              EVP_PKEY *pkey, const EVP_MD *digest,
                              OSSL_LIB_CTX *libctx, const char *propq);

int OSSL_CRMF_MSGS_verify_popo(const OSSL_CRMF_MSGS *reqs,
                               int rid, int acceptRAVerified,
                               OSSL_LIB_CTX *libctx, const char *propq);
```

DESCRIPTION

OSSL_CRMF_MSG_set0_validity() sets the *notBefore* and *notAfter* fields as validity constraints in the certTemplate of *crm*. Any of the *notBefore* and *notAfter* parameters may be NULL, which means no constraint for the respective field. On success ownership of *notBefore* and *notAfter* is transferred to *crm*.

OSSL_CRMF_MSG_set_certReqId() sets *rid* as the certReqId of *crm*.

OSSL_CRMF_CERTTEMPLATE_fill() sets those fields of the certTemplate *tmpl* for which non-NULL values are provided: *pubkey*, *subject*, *issuer*, and/or *serial*. X.509 extensions may be set using **OSSL_CRMF_MSG_set0_extensions()**. On success the reference counter of the *pubkey* (if given) is incremented, while the *subject*, *issuer*, and *serial* structures (if given) are copied.

OSSL_CRMF_MSG_set0_extensions() sets *exts* as the extensions in the certTemplate of *crm*. Frees any pre-existing ones and consumes *exts*.

OSSL_CRMF_MSG_push0_extension() pushes the X509 extension *ext* to the extensions in the certTemplate of *crm*. Consumes *ext*.

OSSL_CRMF_MSG_create_popo() creates and sets the Proof-of-Possession (POPO) according to the method *meth* in *crm*. The library context *libctx* and property query string *propq*, may be NULL to select the defaults. In case the method is OSSL_CRMF_POPO_SIGNATURE the POPO is calculated using the private key *pkey* and the digest method *digest*, where the *digest* argument is ignored if *pkey* is of a type (such as Ed25519 and Ed448) that is implicitly associated with a digest algorithm.

meth can be one of the following:

- OSSL_CRMF_POPO_NONE - RFC 4211, section 4, POP field omitted. CA/RA uses out-of-band method to verify POP. Note that servers may fail in this case, resulting for instance in HTTP error code 500 (Internal error).

- OSSL_CRMF_POPO_RAVERIFIED - RFC 4211, section 4, explicit indication that the RA has already verified the POP.

- OSSL_CRMF_POPO_SIGNATURE - RFC 4211, section 4.1, only case 3 supported so far.

- OSSL_CRMF_POPO_KEYENC - RFC 4211, section 4.2, only indirect method (subsequentMessage/enccert) supported, challenge-response exchange (challengeResp) not yet supported.

- OSSL_CRMF_POPO_KEYAGREE - RFC 4211, section 4.3, not yet supported.

OSSL_CRMF_MSGS_verify_popo verifies the Proof-of-Possession of the request with the given *rid* in the list of *reqs*. Optionally accepts RAVerified. It can make use of the library context *libctx* and property query string *propq*.

RETURN VALUES

All functions return 1 on success, 0 on error.

SEE ALSO

RFC 4211

HISTORY

The OpenSSL CRMF support was added in OpenSSL 3.0.

COPYRIGHT