



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 'OSSL_CMP_exec_KUR_ses.3oss1' command

\$ man OSSL_CMP_exec_KUR_ses.3oss1

OSSL_CMP_EXEC_CERTREQ(3oss1) OpenSSL OSSL_CMP_EXEC_CERTREQ(3oss1)

NAME

OSSL_CMP_exec_certreq, OSSL_CMP_exec_IR_ses, OSSL_CMP_exec_CR_ses,
OSSL_CMP_exec_P10CR_ses, OSSL_CMP_exec_KUR_ses, OSSL_CMP_IR,
OSSL_CMP_CR, OSSL_CMP_P10CR, OSSL_CMP_KUR, OSSL_CMP_try_certreq,
OSSL_CMP_exec_RR_ses, OSSL_CMP_exec_GENM_ses - functions implementing
CMP client transactions

SYNOPSIS

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

```
X509 *OSSL_CMP_exec_certreq(OSSL_CMP_CTX *ctx, int req_type,  
                            const OSSL_CRMF_MSG *crm);
```

```
X509 *OSSL_CMP_exec_IR_ses(OSSL_CMP_CTX *ctx);
```

```
X509 *OSSL_CMP_exec_CR_ses(OSSL_CMP_CTX *ctx);
```

```
X509 *OSSL_CMP_exec_P10CR_ses(OSSL_CMP_CTX *ctx);
```

```
X509 *OSSL_CMP_exec_KUR_ses(OSSL_CMP_CTX *ctx);
```

```
#define OSSL_CMP_IR
```

```
#define OSSL_CMP_CR
```

```
#define OSSL_CMP_P10CR
```

```
#define OSSL_CMP_KUR
```

```
int OSSL_CMP_try_certreq(OSSL_CMP_CTX *ctx, int req_type,
```

```
const OSSL_CRMF_MSG *crm, int *checkAfter);  
int OSSL_CMP_exec_RR_ses(OSSL_CMP_CTX *ctx);  
STACK_OF(OSSL_CMP_ITAV) *OSSL_CMP_exec_GENM_ses(OSSL_CMP_CTX *ctx);
```

DESCRIPTION

This is the OpenSSL API for doing CMP (Certificate Management Protocol) client-server transactions, i.e., sequences of CMP requests and responses.

All functions take a populated OSSL_CMP_CTX structure as their first argument. Usually the server name, port, and path ("CMP alias") need to be set, as well as credentials the client can use for authenticating itself to the client. In order to authenticate the server the client typically needs a trust store. The functions return their respective main results directly, while there are also accessor functions for retrieving various results and status information from the ctx. See OSSL_CMP_CTX_new(3) etc. for details.

The default conveying protocol is HTTP. Timeout values may be given per request-response pair and per transaction. See OSSL_CMP_MSG_http_perform(3) for details.

OSSL_CMP_exec_IR_ses() requests an initial certificate from the given PKI.

OSSL_CMP_exec_CR_ses() requests an additional certificate.

OSSL_CMP_exec_P10CR_ses() conveys a legacy PKCS#10 CSR requesting a certificate.

OSSL_CMP_exec_KUR_ses() obtains an updated certificate.

These four types of certificate enrollment are implemented as macros

calling `OSSL_CMP_exec_certreq()`.

`OSSL_CMP_exec_certreq()` performs a certificate request of the type specified by the `req_type` parameter, which may be IR, CR, P10CR, or KUR. For IR, CR, and KUR, the certificate template to be used in the request may be supplied via the `crm` parameter pointing to a CRMF structure. Typically `crm` is NULL, then the template ingredients are taken from `ctx` and need to be filled in using

`OSSL_CMP_CTX_set1_subjectName(3)`, `OSSL_CMP_CTX_set0_newPkey(3)`, `OSSL_CMP_CTX_set1_oldCert(3)`, etc. For P10CR, `OSSL_CMP_CTX_set1_p10CSR(3)` needs to be used instead. The enrollment session may be blocked by sleeping until the addressed CA (or an intermediate PKI component) can fully process and answer the request.

`OSSL_CMP_try_certreq()` is an alternative to the above functions that is more flexible regarding what to do after receiving a `checkAfter` value. When called for the first time (with no certificate request in progress for the given `ctx`) it starts a new transaction by sending a certificate request constructed as stated above using the `req_type` and optional `crm` parameter. Otherwise (when according to `ctx` a 'waiting' status has been received before) it continues polling for the pending request unless the `req_type` argument is < 0 , which aborts the request. If the requested certificate is available the function returns 1 and the caller can use `OSSL_CMP_CTX_get0_newCert(3)` to retrieve the new certificate. If no error occurred but no certificate is available yet then `OSSL_CMP_try_certreq()` remembers in the CMP context that it should be retried and returns -1 after assigning the received `checkAfter` value via the output pointer argument (unless it is NULL). The `checkAfter` value indicates the number of seconds the caller should let pass before trying again. The caller is free to sleep for the given number of seconds or for some other time and/or to do anything else before retrying by calling `OSSL_CMP_try_certreq()` again with the same parameter values as before. `OSSL_CMP_try_certreq()` then polls to see

whether meanwhile the requested certificate is available. If the caller decides to abort the pending certificate request and provides a negative value as the req_type argument then OSSL_CMP_try_certreq() aborts the CMP transaction by sending an error message to the server.

OSSL_CMP_exec_RR_ses() requests the revocation of the certificate specified in the ctx using OSSL_CMP_CTX_set1_oldCert(3). RFC 4210 is vague in which PKIStatus should be returned by the server. We take "accepted" and "grantedWithMods" as clear success and handle "revocationWarning" and "revocationNotification" just as warnings because CAs typically return them as an indication that the certificate was already revoked. "rejection" is a clear error. The values "waiting" and "keyUpdateWarning" make no sense for revocation and thus are treated as an error as well.

OSSL_CMP_exec_GENM_ses() sends a general message containing the sequence of infoType and infoValue pairs (InfoTypeAndValue; short: ITAV) provided in the ctx using OSSL_CMP_CTX_push0_genm_ITAV(3). It returns the list of ITAVs received in the GenRep. This can be used, for instance, to poll for CRLs or CA Key Updates. See RFC 4210 section 5.3.19 and appendix E.5 for details.

NOTES

CMP is defined in RFC 4210 (and CRMF in RFC 4211).

So far the CMP client implementation is limited to one request per CMP message (and consequently to at most one response component per CMP message).

RETURN VALUES

OSSL_CMP_exec_certreq(), OSSL_CMP_exec_IR_ses(), OSSL_CMP_exec_CR_ses(), OSSL_CMP_exec_P10CR_ses(), and OSSL_CMP_exec_KUR_ses() return a pointer to the newly obtained X509

certificate on success, NULL on error. This pointer will be freed implicitly by `OSSL_CMP_CTX_free()` or `CSSL_CMP_CTX_reinit()`.

`OSSL_CMP_try_certreq()` returns 1 if the requested certificate is available via `OSSL_CMP_CTX_get0_newCert(3)` or on successfully aborting a pending certificate request, 0 on error, and -1 in case a 'waiting' status has been received and `checkAfter` value is available. In the latter case `OSSL_CMP_CTX_get0_newCert(3)` yields NULL and the output parameter `checkAfter` has been used to assign the received value unless `checkAfter` is NULL.

`OSSL_CMP_exec_RR_ses()` returns 1 on success, 0 on error.

`OSSL_CMP_exec_GENM_ses()` returns a pointer to the received ITAV sequence on success, NULL on error. This pointer must be freed by the caller.

EXAMPLES

See `OSSL_CMP_CTX` for examples on how to prepare the context for these functions.

SEE ALSO

`OSSL_CMP_CTX_new(3)`, `OSSL_CMP_CTX_free(3)`,
`OSSL_CMP_CTX_set1_subjectName(3)`, `OSSL_CMP_CTX_set0_newPkey(3)`,
`OSSL_CMP_CTX_set1_p10CSR(3)`, `OSSL_CMP_CTX_set1_oldCert(3)`,
`OSSL_CMP_CTX_get0_newCert(3)`, `OSSL_CMP_CTX_push0_genm_ITAV(3)`,
`OSSL_CMP_MSG_http_perform(3)`

HISTORY

The OpenSSL CMP support was added in OpenSSL 3.0.

COPYRIGHT

Copyright 2007-2021 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 OSSL_CMP_EXEC_CERTREQ(3ossl)