NAME

OSSL_CMP_SRV_process_request, OSSL_CMP_CTX_server_perform, OSSL_CMP_SRV_CTX_new, OSSL_CMP_SRV_CTX_free, OSSL_CMP_SRV_cert_request_cb_t, OSSL_CMP_SRV_rr_cb_t, OSSL_CMP_SRV_certConf_cb_t, OSSL_CMP_SRV_genm_cb_t, OSSL_CMP_SRV_error_cb_t, OSSL_CMP_SRV_pollReq_cb_t, OSSL_CMP_SRV_CTX_init, OSSL_CMP_SRV_CTX_get0_cmp_ctx, OSSL_CMP_SRV_CTX_get0_custom_ctx, OSSL_CMP_SRV_CTX_set_send_unprotected_errors, OSSL_CMP_SRV_CTX_set_accept_unprotected, OSSL_CMP_SRV_CTX_set_accept_raverified, OSSL_CMP_SRV_CTX_set_grant_implicit_confirm - generic functions to set up and control a CMP server

SYNOPSIS

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

OSSL_CMP_MSG *OSSL_CMP_SRV_process_request(OSSL_CMP_SRV_CTX *srv_ctx,
                                           const OSSL_CMP_MSG *req);
OSSL_CMP_MSG *OSSL_CMP_CTX_server_perform(OSSL_CMP_CTX *client_ctx,
                                          const OSSL_CMP_MSG *req);
OSSL_CMP_SRV_CTX *OSSL_CMP_SRV_CTX_new(OSSL_LIB_CTX *libctx, const char *propq);
void OSSL_CMP_SRV_CTX_free(OSSL_CMP_SRV_CTX *srv_ctx);

typedef OSSL_CMP_PKISI *(*OSSL_CMP_SRV_cert_request_cb_t)(
                                                OSSL_CMP_SRV_CTX *srv_ctx,
                                                const OSSL_CMP_MSG *req,
                                                int certReqId,
                                                const OSSL_CRMF_MSG *crm,
                                                const X509_REQ *p10cr,
                                                X509 **certOut,
                                                STACK_OF(X509) **chainOut,
                                                STACK_OF(X509) **caPubs);
typedef OSSL_CMP_PKISI *(*OSSL_CMP_SRV_rr_cb_t)(OSSL_CMP_SRV_CTX *srv_ctx,
                                                const OSSL_CMP_MSG *req,
                                                const X509_NAME *issuer,
                                                const ASN1_INTEGER *serial);
typedef int (*OSSL_CMP_SRV_genm_cb_t)(OSSL_CMP_SRV_CTX *srv_ctx,
                                      const OSSL_CMP_MSG *req,
                                      STACK_OF(OSSL_CMP_ITAV) *in,
                                      STACK_OF(OSSL_CMP_ITAV) **out);
typedef void (*OSSL_CMP_SRV_error_cb_t)(OSSL_CMP_SRV_CTX *srv_ctx,
                                        const OSSL_CMP_MSG *req,
                                        const OSSL_CMP_PKISI *statusInfo,
                                        const ASN1_INTEGER *errorCode,
                                        const OSSL_CMP_PKIFREETEXT *errorDetails);
typedef int (*OSSL_CMP_SRV_certConf_cb_t)(OSSL_CMP_SRV_CTX *srv_ctx,
                                          const OSSL_CMP_MSG *req,
                                          int certReqId,
                                          const ASN1_OCTET_STRING *certHash,
                                          const OSSL_CMP_PKISI *si);
typedef int (*OSSL_CMP_SRV_pollReq_cb_t)(OSSL_CMP_SRV_CTX *srv_ctx,
                                         const OSSL_CMP_MSG *req,
                                         int certReqId,
                                         OSSL_CMP_MSG **certReq,
                                         int64_t *check_after);
int OSSL_CMP_SRV_CTX_init(OSSL_CMP_SRV_CTX *srv_ctx, void *custom_ctx,
                          OSSL_CMP_SRV_cert_request_cb_t process_cert_request,
                          OSSL_CMP_SRV_rr_cb_t process_rr,
                          OSSL_CMP_SRV_genm_cb_t process_genm,
                          OSSL_CMP_SRV_error_cb_t process_error,
                          OSSL_CMP_SRV_certConf_cb_t process_certConf,
                          OSSL_CMP_SRV_pollReq_cb_t process_pollReq);

OSSL_CMP_CTX *OSSL_CMP_SRV_CTX_get0_cmp_ctx(const OSSL_CMP_SRV_CTX *srv_ctx);
void *OSSL_CMP_SRV_CTX_get0_custom_ctx(const OSSL_CMP_SRV_CTX *srv_ctx);

int OSSL_CMP_SRV_CTX_set_send_unprotected_errors(OSSL_CMP_SRV_CTX *srv_ctx,
                                                 int val);
int OSSL_CMP_SRV_CTX_set_accept_unprotected(OSSL_CMP_SRV_CTX *srv_ctx, int val);
int OSSL_CMP_SRV_CTX_set_accept_raverified(OSSL_CMP_SRV_CTX *srv_ctx, int val);
int OSSL_CMP_SRV_CTX_set_grant_implicit_confirm(OSSL_CMP_SRV_CTX *srv_ctx,
                                                int val);
```

DESCRIPTION

OSSL_CMP_SRV_process_request() implements the generic aspects of a CMP server. Its arguments are the **OSSL_CMP_SRV_CTX** *srv_ctx* and the CMP request message *req*. It does the typical generic checks on *req*, calls the respective callback function (if present) for more specific processing, and then assembles a result message, which may be a CMP error message. If after return of the function the expression *OSSL_CMP_CTX_get_status(OSSL_CMP_SRV_CTX_get0_cmp_ctx(srv_ctx))* yields -1 then the function has closed the current transaction, which may be due to normal successful end of the transaction or due to an error.

OSSL_CMP_CTX_server_perform() is an interface to **OSSL_CMP_SRV_process_request()** that can be used by a CMP client in the same way as **OSSL_CMP_MSG_http_perform**(3). The **OSSL_CMP_SRV_CTX** must be set as *transfer_cb_arg* of *client_ctx*.

OSSL_CMP_SRV_CTX_new() creates and initializes an **OSSL_CMP_SRV_CTX** structure associated with the library context *libctx* and property query string *propq*, both of which may be NULL to select the defaults.

OSSL_CMP_SRV_CTX_free() deletes the given *srv_ctx*.

OSSL_CMP_SRV_CTX_init() sets in the given *srv_ctx* a custom server context pointer as well as callback functions performing the specific processing of CMP certificate requests, revocation requests, certificate confirmation requests, general messages, error messages, and poll requests. All arguments except *srv_ctx* may be NULL. If a callback for some message type is not given this means that the respective type of CMP message is not supported by the server.

OSSL_CMP_SRV_CTX_get0_cmp_ctx() returns the **OSSL_CMP_CTX** from the *srv_ctx*.

OSSL_CMP_SRV_CTX_get0_custom_ctx() returns the custom server context from *srv_ctx* that has been set using **OSSL_CMP_SRV_CTX_init()**.

OSSL_CMP_SRV_CTX_set_send_unprotected_errors() enables sending error messages and other forms of negative responses unprotected.

OSSL_CMP_SRV_CTX_set_accept_unprotected() enables acceptance of requests without protection of with invalid protection.

OSSL_CMP_SRV_CTX_set_accept_raverified() enables acceptance of ir/cr/kur messages with POPO ’RAVerified’.

OSSL_CMP_SRV_CTX_set_grant_implicit_confirm() enables granting implicit confirmation of newly enrolled certificates if requested.

NOTES

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

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

RETURN VALUES

OSSL_CMP_SRV_CTX_new() returns a **OSSL_CMP_SRV_CTX** structure on success, NULL on error.

OSSL_CMP_SRV_CTX_free() does not return a value.

OSSL_CMP_SRV_CTX_get0_cmp_ctx() returns a **OSSL_CMP_CTX** structure on success, NULL on error.

OSSL_CMP_SRV_CTX_get0_custom_ctx() returns the custom server context that has been set using **OSSL_CMP_SRV_CTX_init()**.

All other functions return 1 on success, 0 on error.

HISTORY

The OpenSSL CMP support was added in OpenSSL 3.0.

COPYRIGHT