Rocky Enterprise Linux 9.2 Manual Pages on command 'CT_POLICY_EVAL_CTX_get0_cert.3ossl'

$ man CT_POLICY_EVAL_CTX_get0_cert.3ossl

CT_POLICY_EVAL_CTX_NEW(3ossl) OpenSSL CT_POLICY_EVAL_CTX_NEW(3ossl)

NAME

CT_POLICY_EVAL_CTX_new_ex, CT_POLICY_EVAL_CTX_new,

CT_POLICY_EVAL_CTX_free, CT_POLICY_EVAL_CTX_get0_cert,

CT_POLICY_EVAL_CTX_set1_cert, CT_POLICY_EVAL_CTX_get0_issuer,

CT_POLICY_EVAL_CTX_set1_issuer, CT_POLICY_EVAL_CTX_get0_log_store,

CT_POLICY_EVAL_CTX_set_shared_CTLOG_STORE, CT_POLICY_EVAL_CTX_get_time,

CT_POLICY_EVAL_CTX_set_time - Encapsulates the data required to

evaluate whether SCTs meet a Certificate Transparency policy

SYNOPSIS

#include <openssl/ct.h>

CT_POLICY_EVAL_CTX *CT_POLICY_EVAL_CTX_new_ex(OSSL_LIB_CTX *libctx,

const char *propq);

CT_POLICY_EVAL_CTX *CT_POLICY_EVAL_CTX_new(void);

void CT_POLICY_EVAL_CTX_free(CT_POLICY_EVAL_CTX *ctx);

```
X509* CT_POLICY_EVAL_CTX_get0_cert(const CT_POLICY_EVAL_CTX *ctx);
int CT_POLICY_EVAL_CTX_set1_cert(CT_POLICY_EVAL_CTX *ctx, X509 *cert);
X509* CT_POLICY_EVAL_CTX_get0_issuer(const CT_POLICY_EVAL_CTX *ctx);
int CT_POLICY_EVAL_CTX_set1_issuer(CT_POLICY_EVAL_CTX *ctx, X509 *issuer);
const CTLOG_STORE *CT_POLICY_EVAL_CTX_get0_log_store(const CT_POLICY_EVAL_CTX *ctx);
void CT_POLICY_EVAL_CTX_set_shared_CTLOG_STORE(CT_POLICY_EVAL_CTX *ctx,
                                               CTLOG_STORE *log_store);
uint64_t CT_POLICY_EVAL_CTX_get_time(const CT_POLICY_EVAL_CTX *ctx);
void CT_POLICY_EVAL_CTX_set_time(CT_POLICY_EVAL_CTX *ctx, uint64_t time_in_ms);
```

DESCRIPTION

A CT_POLICY_EVAL_CTX is used by functions that evaluate whether Signed Certificate Timestamps (SCTs) fulfil a Certificate Transparency (CT) policy. This policy may be, for example, that at least one valid SCT is available. To determine this, an SCT's timestamp and signature must be verified. This requires:

? the public key of the log that issued the SCT

? the certificate that the SCT was issued for

? the issuer certificate (if the SCT was issued for a pre-certificate)

? the current time

The above requirements are met using the setters described below.

CT_POLICY_EVAL_CTX_new_ex() creates an empty policy evaluation context and associates it with the given library context libctx and property query string propq.

CT_POLICY_EVAL_CTX_new() does the same thing as CT_POLICY_EVAL_CTX_new_ex() except that it uses the default library

context and property query string.

The CT_POLICY_EVAL_CTX should then be populated using:

? CT_POLICY_EVAL_CTX_set1_cert() to provide the certificate the SCTs
were issued for

Increments the reference count of the certificate.

? CT_POLICY_EVAL_CTX_set1_issuer() to provide the issuer certificate

Increments the reference count of the certificate.

? CT_POLICY_EVAL_CTX_set_shared_CTLOG_STORE() to provide a list of logs
that are trusted as sources of SCTs

Holds a pointer to the CTLOG_STORE, so the CTLOG_STORE must outlive
the CT_POLICY_EVAL_CTX.

? CT_POLICY_EVAL_CTX_set_time() to set the time SCTs should be compared
with to determine if they are valid

The SCT timestamp will be compared to this time to check whether the
SCT was issued in the future. RFC6962 states that "TLS clients MUST
reject SCTs whose timestamp is in the future". By default, this will
be set to 5 minutes in the future (e.g. (time() + 300) * 1000), to
allow for clock drift.

The time should be in milliseconds since the Unix Epoch.

Each setter has a matching getter for accessing the current value.

When no longer required, the CT_POLICY_EVAL_CTX should be passed to

CT_POLICY_EVAL_CTX_free() to delete it.

NOTES

The issuer certificate only needs to be provided if at least one of the SCTs was issued for a pre-certificate. This will be the case for SCTs embedded in a certificate (i.e. those in an X.509 extension), but may not be the case for SCTs found in the TLS SCT extension or OCSP response.

RETURN VALUES

CT_POLICY_EVAL_CTX_new_ex() and CT_POLICY_EVAL_CTX_new() will return NULL if malloc fails.

SEE ALSO

ct(7)

HISTORY

CT_POLICY_EVAL_CTX_new_ex was added in OpenSSL 3.0. All other functions were added in OpenSSL 1.1.0.

COPYRIGHT