



*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 'CT\_POLICY\_EVAL\_CTX\_new\_ex.3oss!' command**

**\$ man CT\_POLICY\_EVAL\_CTX\_new\_ex.3oss!**

CT\_POLICY\_EVAL\_CTX\_NEW(3oss!)    OpenSSL    CT\_POLICY\_EVAL\_CTX\_NEW(3oss!)

### 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.  $(\text{time}() + 300) * 1000$ ), 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

Copyright 2016-2020 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    CT\_POLICY\_EVAL\_CTX\_NEW(3ossl)