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 'OCSP_resp_get0_signer.3ossl' command

$ man OCSP_resp_get0_signer.3ossl

OCSP_RESP_FIND_STATUS(3ossl) OpenSSL OCSP_RESP_FIND_STATUS(3ossl)

NAME

OCSP_resp_find_status, OCSP_resp_count, OCSP_resp_get0, OCSP_resp_find,

OCSP_single_get0_status, OCSP_resp_get0_produced_at,

OCSP_resp_get0_signature, OCSP_resp_get0_tbs_sigalg,

OCSP_resp_get0_respdata, OCSP_resp_get0_certs, OCSP_resp_get0_signer,

OCSP_resp_get0_id, OCSP_resp_get1_id, OCSP_check_validity,

OCSP_basic_verify - OCSP response utility functions

SYNOPSIS

```
#include <openssl/ocsp.h>
int OCSP_resp_find_status(OCSP_BASICRESP *bs, OCSP_CERTID *id, int *status,
                          int *reason,
                          ASN1_GENERALIZEDTIME **revtime,
                          ASN1_GENERALIZEDTIME **thisupd,
                          ASN1_GENERALIZEDTIME **nextupd);
int OCSP_resp_count(OCSP_BASICRESP *bs);
OCSP_SINGLERESP *OCSP_resp_get0(OCSP_BASICRESP *bs, int idx);
int OCSP_resp_find(OCSP_BASICRESP *bs, OCSP_CERTID *id, int last);
int OCSP_single_get0_status(OCSP_SINGLERESP *single, int *reason,
                            ASN1_GENERALIZEDTIME **revtime,
                            ASN1_GENERALIZEDTIME **thisupd,
                            ASN1_GENERALIZEDTIME **nextupd);
const ASN1_GENERALIZEDTIME *OCSP_resp_get0_produced_at(
```

```
                        const OCSP_BASICRESP* single);
    const ASN1_OCTET_STRING *OCSP_resp_get0_signature(const OCSP_BASICRESP *bs);
    const X509_ALGOR *OCSP_resp_get0_tbs_sigalg(const OCSP_BASICRESP *bs);
    const OCSP_RESPDATA *OCSP_resp_get0_respdata(const OCSP_BASICRESP *bs);
    const STACK_OF(X509) *OCSP_resp_get0_certs(const OCSP_BASICRESP *bs);
    int OCSP_resp_get0_signer(OCSP_BASICRESP *bs, X509 **signer,
                          STACK_OF(X509) *extra_certs);
    int OCSP_resp_get0_id(const OCSP_BASICRESP *bs,
                      const ASN1_OCTET_STRING **pid,
                      const X509_NAME **pname);
    int OCSP_resp_get1_id(const OCSP_BASICRESP *bs,
                      ASN1_OCTET_STRING **pid,
                      X509_NAME **pname);
    int OCSP_check_validity(ASN1_GENERALIZEDTIME *thisupd,
                        ASN1_GENERALIZEDTIME *nextupd,
                        long sec, long maxsec);
    int OCSP_basic_verify(OCSP_BASICRESP *bs, STACK_OF(X509) *certs,
                      X509_STORE *st, unsigned long flags);
```

DESCRIPTION

OCSP_resp_find_status() searches bs for an OCSP response for id. If it is successful the fields of the response are returned in *status, *reason, *revtime, *thisupd and *nextupd. The *status value will be one of V_OCSP_CERTSTATUS_GOOD, V_OCSP_CERTSTATUS_REVOKED or V_OCSP_CERTSTATUS_UNKNOWN. The *reason and *revtime fields are only set if the status is V_OCSP_CERTSTATUS_REVOKED. If set the *reason field will be set to the revocation reason which will be one of OCSP_REVOKED_STATUS_NOSTATUS, OCSP_REVOKED_STATUS_UNSPECIFIED, OCSP_REVOKED_STATUS_KEYCOMPROMISE, OCSP_REVOKED_STATUS_CACOMPROMISE, OCSP_REVOKED_STATUS_AFFILIATIONCHANGED, OCSP_REVOKED_STATUS_SUPERSEDED, OCSP_REVOKED_STATUS_CESSATIONOFOPERATION, OCSP_REVOKED_STATUS_CERTIFICATEHOLD or OCSP_REVOKED_STATUS_REMOVEFROMCRL.

OCSP_resp_count() returns the number of OCSP_SINGLERESP structures in

bs.

OCSP_resp_get0() returns the OCSP_SINGLERESP structure in bs corresponding to index idx, where idx runs from 0 to OCSP_resp_count(bs) - 1.

OCSP_resp_find() searches bs for id and returns the index of the first matching entry after last or starting from the beginning if last is -1.

OCSP_single_get0_status() extracts the fields of single in *reason, *revtime, *thisupd and *nextupd.

OCSP_resp_get0_produced_at() extracts the producedAt field from the single response bs.

OCSP_resp_get0_signature() returns the signature from bs.

OCSP_resp_get0_tbs_sigalg() returns the signatureAlgorithm from bs.

OCSP_resp_get0_respdata() returns the tbsResponseData from bs.

OCSP_resp_get0_certs() returns any certificates included in bs.

OCSP_resp_get0_signer() attempts to retrieve the certificate that directly signed bs. The OCSP protocol does not require that this certificate is included in the certs field of the response, so additional certificates can be supplied via the extra_certs if the certificates that may have signed the response are known via some out-of-band mechanism.

OCSP_resp_get0_id() gets the responder id of bs. If the responder ID is a name then <*pname> is set to the name and *pid is set to NULL. If the responder ID is by key ID then *pid is set to the key ID and *pname is set to NULL.

OCSP_resp_get1_id() is the same as OCSP_resp_get0_id() but leaves ownership of *pid and *pname with the caller, who is responsible for freeing them unless the function returns 0.

OCSP_check_validity() checks the validity of its thisupd and nextupd arguments, which will be typically obtained from OCSP_resp_find_status() or OCSP_single_get0_status(). If sec is nonzero it indicates how many seconds leeway should be allowed in the check. If maxsec is positive it indicates the maximum age of thisupd in seconds.

OCSP_basic_verify() checks that the basic response message bs is

correctly signed and that the signer certificate can be validated. It takes st as the trusted store and certs as a set of untrusted intermediate certificates. The function first tries to find the signer certificate of the response in certs. It then searches the certificates the responder may have included in bs unless flags contains OCSP_NOINTERN. It fails if the signer certificate cannot be found. Next, unless flags contains OCSP_NOSIGS, the function checks the signature of bs and fails on error. Then the function already returns success if flags contains OCSP_NOVERIFY or if the signer certificate was found in certs and flags contains OCSP_TRUSTOTHER. Otherwise the function continues by validating the signer certificate. If flags contains OCSP_PARTIAL_CHAIN it takes intermediate CA certificates in st as trust anchors. For more details, see the description of X509_V_FLAG_PARTIAL_CHAIN in "VERIFICATION FLAGS" in X509_VERIFY_PARAM_set_flags(3). If flags contains OCSP_NOCHAIN it ignores all certificates in certs and in bs, else it takes them as untrusted intermediate CA certificates and uses them for constructing the validation path for the signer certificate. Certicate revocation status checks using CRLs is disabled during path validation if the signer certificate contains the id-pkix-ocsp-no-check extension. After successful path validation the function returns success if the OCSP_NOCHECKS flag is set. Otherwise it verifies that the signer certificate meets the OCSP issuer criteria including potential delegation. If this does not succeed and the OCSP_NOEXPLICIT flag is not set the function checks for explicit trust for OCSP signing in the root CA certificate.

RETURN VALUES

OCSP_resp_find_status() returns 1 if id is found in bs and 0 otherwise.

OCSP_resp_count() returns the total number of OCSP_SINGLERESP fields in bs or -1 on error.

OCSP_resp_get0() returns a pointer to an OCSP_SINGLERESP structure or NULL on error, such as idx being out of range.

OCSP_resp_find() returns the index of id in bs (which may be 0) or -1

on error, such as when id was not found.

OCSP_single_get0_status() returns the status of single or -1 if an error occurred.

OCSP_resp_get0_produced_at() returns the producedAt field from bs.

OCSP_resp_get0_signature() returns the signature from bs.

OCSP_resp_get0_tbs_sigalg() returns the signatureAlgorithm field from bs.

OCSP_resp_get0_respdata() returns the tbsResponseData field from bs.

OCSP_resp_get0_certs() returns any certificates included in bs.

OCSP_resp_get0_signer() returns 1 if the signing certificate was located, or 0 if not found or on error.

OCSP_resp_get0_id() and OCSP_resp_get1_id() return 1 on success, 0 on failure.

OCSP_check_validity() returns 1 if thisupd and nextupd are valid time values and the current time + sec is not before thisupd and, if maxsec >= 0, the current time - maxsec is not past nextupd. Otherwise it returns 0 to indicate an error.

OCSP_basic_verify() returns 1 on success, 0 on verification not successful, or -1 on a fatal error such as malloc failure.

NOTES

Applications will typically call OCSP_resp_find_status() using the certificate ID of interest and then check its validity using OCSP_check_validity(). They can then take appropriate action based on the status of the certificate.

An OCSP response for a certificate contains thisUpdate and nextUpdate fields. Normally the current time should be between these two values.

To account for clock skew the maxsec field can be set to nonzero in OCSP_check_validity(). Some responders do not set the nextUpdate field, this would otherwise mean an ancient response would be considered valid: the maxsec parameter to OCSP_check_validity() can be used to limit the permitted age of responses.

The values written to *revtime, *thisupd and *nextupd by OCSP_resp_find_status() and OCSP_single_get0_status() are internal

pointers which MUST NOT be freed up by the calling application. Any or all of these parameters can be set to NULL if their value is not required.

SEE ALSO

crypto(7), OCSP_cert_to_id(3), OCSP_request_add1_nonce(3), OCSP_REQUEST_new(3), OCSP_response_status(3), OCSP_sendreq_new(3), X509_VERIFY_PARAM_set_flags(3)

COPYRIGHT