NAME

gnutls_certificate_verify_peers - API function

SYNOPSIS

#include <gnutls/gnutls.h>

int gnutls_certificate_verify_peers(gnutls_session_t *session***, gnutls_typed_vdata_st *** *data***, unsigned int** *elements***, unsigned int *** *status***);**

ARGUMENTS

gnutls_session_t session
: is a gnutls session

gnutls_typed_vdata_st * data
: an array of typed data

unsigned int elements
: the number of data elements

unsigned int * status
: is the output of the verification

DESCRIPTION

This function will verify the peer's certificate and store the the status in the *status* variable as a bitwise OR of gnutls_certificate_status_t values or zero if the certificate is trusted. Note that value in *status* is set only when the return value of this function is success (i.e, failure to trust a certificate does not imply a negative return value). The default verification flags used by this function can be overridden using **gnutls_certificate_set_verify_flags()**. See the documentation of **gnutls_certificate_verify_peers2()** for details in the verification process.

This function will take into account the stapled OCSP responses sent by the server, as well as the following X.509 certificate extensions: Name Constraints, Key Usage, and Basic Constraints (pathlen).

The acceptable *data* types are **GNUTLS_DT_DNS_HOSTNAME**, **GNUTLS_DT_RFC822NAME** and **GNUTLS_DT_KEY_PURPOSE_OID**. The former two accept as data a null-terminated hostname or email address, and the latter a null-terminated object identifier (e.g., **GNUTLS_KP_TLS_WWW_SERVER**).

If a DNS hostname is provided then this function will compare the hostname in the certificate against the given. If names do not match the **GNUTLS_CERT_UNEXPECTED_OWNER** status flag will be

set. If a key purpose OID is provided and the end-certificate contains the extended key usage PKIX extension, it will be required to be have the provided key purpose or be marked for any purpose, otherwise verification status will have the **GNUTLS_CERT_SIGNER_CONSTRAINTS_FAILURE** flag set.

To avoid denial of service attacks some default upper limits regarding the certificate key size and chain size are set. To override them use **gnutls_certificate_set_verify_limits()**.

Note that when using raw public-keys verification will not work because there is no corresponding certificate body belonging to the raw key that can be verified. In that case this function will return **GNUTLS_E_INVALID_REQUEST**.

RETURNS

GNUTLS_E_SUCCESS (0) when the validation is performed, or a negative error code otherwise. A successful error code means that the *status* parameter must be checked to obtain the validation status.

SINCE

3.3.0

REPORTING BUGS

Report bugs to <bugs@gnutls.org>.
Home page: https://www.gnutls.org

SEE ALSO

The full documentation for **gnutls** is maintained as a Texinfo manual. If the /usr/local/share/doc/gnutls/ directory does not contain the HTML form visit

https://www.gnutls.org/manual/