NAME

SSL_CTX_set_session_ticket_cb, SSL_SESSION_get0_ticket_appdata, SSL_SESSION_set1_ticket_appdata, SSL_CTX_generate_session_ticket_fn, SSL_CTX_decrypt_session_ticket_fn - manage session ticket application data

SYNOPSIS

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

typedef int (*SSL_CTX_generate_session_ticket_fn)(SSL *s, void *arg);
typedef SSL_TICKET_RETURN (*SSL_CTX_decrypt_session_ticket_fn)(SSL *s, SSL_SESSION *ss,
                                                const unsigned char *keyname,
                                                size_t keyname_len,
                                                SSL_TICKET_STATUS status,
                                                void *arg);
int SSL_CTX_set_session_ticket_cb(SSL_CTX *ctx,
                                  SSL_CTX_generate_session_ticket_fn gen_cb,
                                  SSL_CTX_decrypt_session_ticket_fn dec_cb,
                                  void *arg);
int SSL_SESSION_set1_ticket_appdata(SSL_SESSION *ss, const void *data, size_t len);
int SSL_SESSION_get0_ticket_appdata(SSL_SESSION *ss, void **data, size_t *len);
```

DESCRIPTION

SSL_CTX_set_set_session_ticket_cb() sets the application callbacks **gen_cb** and **dec_cb** that are used by a server to set and get application data stored with a session, and placed into a session ticket. Either callback function may be set to NULL. The value of **arg** is passed to the callbacks.

gen_cb is the application defined callback invoked when a session ticket is about to be created. The application can call **SSL_SESSION_set1_ticket_appdata()** at this time to add application data to the session ticket. The value of **arg** is the same as that given to **SSL_CTX_set_session_ticket_cb()**. The **gen_cb** callback is defined as type **SSL_CTX_generate_session_ticket_fn**.

dec_cb is the application defined callback invoked after session ticket decryption has been attempted and any session ticket application data is available. If ticket decryption was successful then the **ss** argument contains the session data. The **keyname** and **keyname_len** arguments identify the key used to decrypt the session ticket. The **status** argument is the result of the ticket decryption. See the "NOTES" section below for further details. The value of **arg** is the same as that given to **SSL_CTX_set_session_ticket_cb()**. The **dec_cb** callback is defined as type **SSL_CTX_decrypt_session_ticket_fn**.

SSL_SESSION_set1_ticket_appdata() sets the application data specified by **data** and **len** into **ss** which

is then placed into any generated session tickets. It can be called at any time before a session ticket is created to update the data placed into the session ticket. However, given that sessions and tickets are created by the handshake, the **gen_cb** is provided to notify the application that a session ticket is about to be generated.

SSL_SESSION_get0_ticket_appdata() assigns **data** to the session ticket application data and assigns **len** to the length of the session ticket application data from **ss**. The application data can be set via **SSL_SESSION_set1_ticket_appdata()** or by a session ticket. NULL will be assigned to **data** and 0 will be assigned to **len** if there is no session ticket application data. **SSL_SESSION_get0_ticket_appdata()** can be called any time after a session has been created. The **dec_cb** is provided to notify the application that a session ticket has just been decrypted.

NOTES

When the **dec_cb** callback is invoked, the SSL_SESSION **ss** has not yet been assigned to the SSL **s**. The **status** indicates the result of the ticket decryption. The callback must check the **status** value before performing any action, as it is called even if ticket decryption fails.

The **keyname** and **keyname_len** arguments to **dec_cb** may be used to identify the key that was used to encrypt the session ticket.

The **status** argument can be any of these values:

SSL_TICKET_EMPTY
: Empty ticket present. No ticket data will be used and a new ticket should be sent to the client. This only occurs in TLSv1.2 or below. In TLSv1.3 it is not valid for a client to send an empty ticket.

SSL_TICKET_NO_DECRYPT
: The ticket couldn’t be decrypted. No ticket data will be used and a new ticket should be sent to the client.

SSL_TICKET_SUCCESS
: A ticket was successfully decrypted, any session ticket application data should be available. A new ticket should not be sent to the client.

SSL_TICKET_SUCCESS_RENEW
: Same as **SSL_TICKET_SUCCESS**, but a new ticket should be sent to the client.

The return value can be any of these values:

SSL_TICKET_RETURN_ABORT

The handshake should be aborted, either because of an error or because of some policy. Note that in TLSv1.3 a client may send more than one ticket in a single handshake. Therefore, just because one ticket is unacceptable it does not mean that all of them are. For this reason this option should be used with caution.

SSL_TICKET_RETURN_IGNORE
Do not use a ticket (if one was available). Do not send a renewed ticket to the client.

SSL_TICKET_RETURN_IGNORE_RENEW
Do not use a ticket (if one was available). Send a renewed ticket to the client.

If the callback does not wish to change the default ticket behaviour then it should return this value if **status** is **SSL_TICKET_EMPTY** or **SSL_TICKET_NO_DECRYPT**.

SSL_TICKET_RETURN_USE
Use the ticket. Do not send a renewed ticket to the client. It is an error for the callback to return this value if **status** has a value other than **SSL_TICKET_SUCCESS** or **SSL_TICKET_SUCCESS_RENEW**.

If the callback does not wish to change the default ticket behaviour then it should return this value if **status** is **SSL_TICKET_SUCCESS**.

SSL_TICKET_RETURN_USE_RENEW
Use the ticket. Send a renewed ticket to the client. It is an error for the callback to return this value if **status** has a value other than **SSL_TICKET_SUCCESS** or **SSL_TICKET_SUCCESS_RENEW**.

If the callback does not wish to change the default ticket behaviour then it should return this value if **status** is **SSL_TICKET_SUCCESS_RENEW**.

If **status** has the value **SSL_TICKET_EMPTY** or **SSL_TICKET_NO_DECRYPT** then no session data will be available and the callback must not use the **ss** argument. If **status** has the value **SSL_TICKET_SUCCESS** or **SSL_TICKET_SUCCESS_RENEW** then the application can call **SSL_SESSION_get0_ticket_appdata()** using the session provided in the **ss** argument to retrieve the application data.

When the **gen_cb** callback is invoked, the **SSL_get_session()** function can be used to retrieve the SSL_SESSION for **SSL_SESSION_set1_ticket_appdata()**.

By default, in TLSv1.2 and below, a new session ticket is not issued on a successful resumption and therefore **gen_cb** will not be called. In TLSv1.3 the default behaviour is to always issue a new ticket on

resumption. In both cases this behaviour can be changed if a ticket key callback is in use (see **SSL_CTX_set_tlsext_ticket_key_cb**(3)).

RETURN VALUES

The **SSL_CTX_set_session_ticket_cb()**, **SSL_SESSION_set1_ticket_appdata()** and **SSL_SESSION_get0_ticket_appdata()** functions return 1 on success and 0 on failure.

The **gen_cb** callback must return 1 to continue the connection. A return of 0 will terminate the connection with an INTERNAL_ERROR alert.

The **dec_cb** callback must return a value as described in "NOTES" above.

SEE ALSO

ssl(7), **SSL_get_session**(3)

HISTORY

The **SSL_CTX_set_session_ticket_cb()**, **SSL_SESSION_set1_ticket_appdata()** and **SSL_SESSION_get_ticket_appdata()** functions were added in OpenSSL 1.1.1.

COPYRIGHT

Copyright 2017-2020 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>.