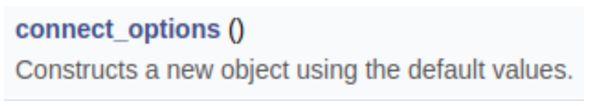
① mqtt::connect_options

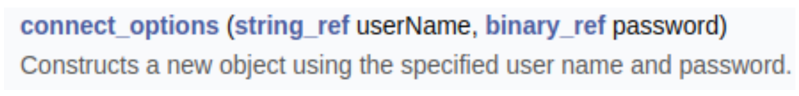
⇛ Holds the set of options that control how the client connects to a server.

★ Public member function

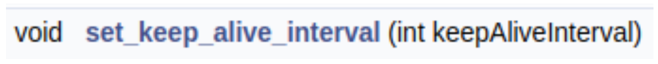
{Default Constructor}



connect_options ()
Constructs a new object using the default values.



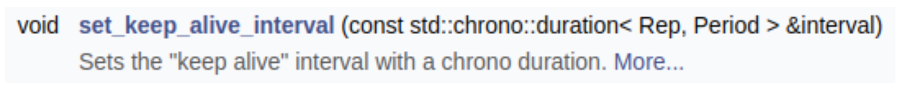
connect_options (**string_ref** userName, **binary_ref** password)
Constructs a new object using the specified user name and password.



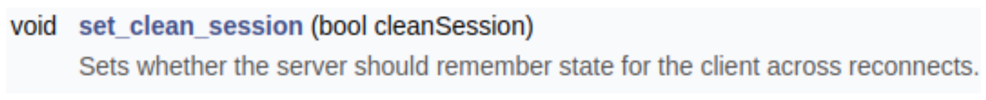
void **set_keep_alive_interval** (int keepAliveInterval)

⇛ This is the maximum time that should pass without communications between client and server. If no massages pass in this time, the client will ping the broker.

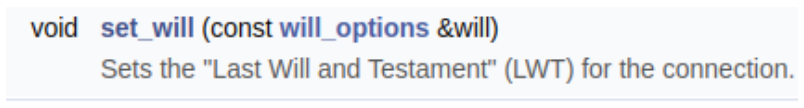
⇛ The keep alive interval in seconds.



void **set_keep_alive_interval** (const std::chrono::duration< Rep, Period > &interval)
Sets the "keep alive" interval with a chrono duration. More...



void **set_clean_session** (bool cleanSession)
Sets whether the server should remember state for the client across reconnects.



void **set_will** (const **will_options** &will)
Sets the "Last Will and Testament" (LWT) for the connection.

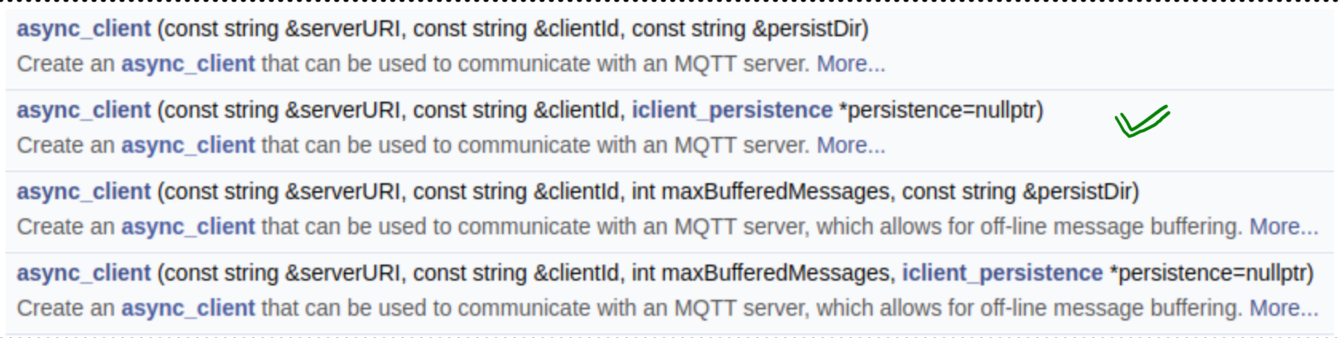
② mqtt::async_client

⇛ Lightweight client for talking to an MQTT server using non-blocking methods that allow an operation to run in the background.

⇛ It is inside

#include <async_client.h>

★ Public member functions



async_client (const string &serverURI, const string &clientId, const string &persistDir)
Create an **async_client** that can be used to communicate with an MQTT server. More...

async_client (const string &serverURI, const string &clientId, **iclient_persistence** *persistence=nullptr) ✓
Create an **async_client** that can be used to communicate with an MQTT server. More...

async_client (const string &serverURI, const string &clientId, int maxBufferedMessages, const string &persistDir)
Create an **async_client** that can be used to communicate with an MQTT server, which allows for off-line message buffering. More...

async_client (const string &serverURI, const string &clientId, int maxBufferedMessages, **iclient_persistence** *persistence=nullptr)
Create an **async_client** that can be used to communicate with an MQTT server, which allows for off-line message buffering. More...

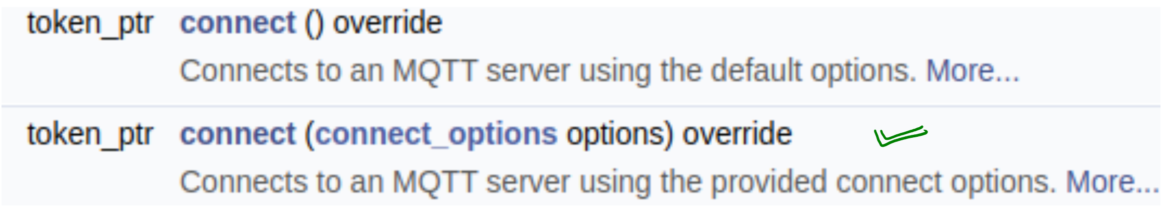
URI {Uniform resource Identifier}

→ It is a string of characters that unambiguously identifies a particular resource.

→ Example: $\underbrace{\text{tcp}}_{\text{Scheme}}\text{://}\underbrace{\text{192.168.43.205}}_{\text{Ip address}}\text{:}\underbrace{\text{1883}}_{\text{Port number}}$

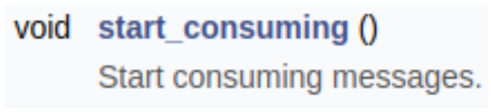
ClientId

→ A client identifier that is unique on the server being connected to.



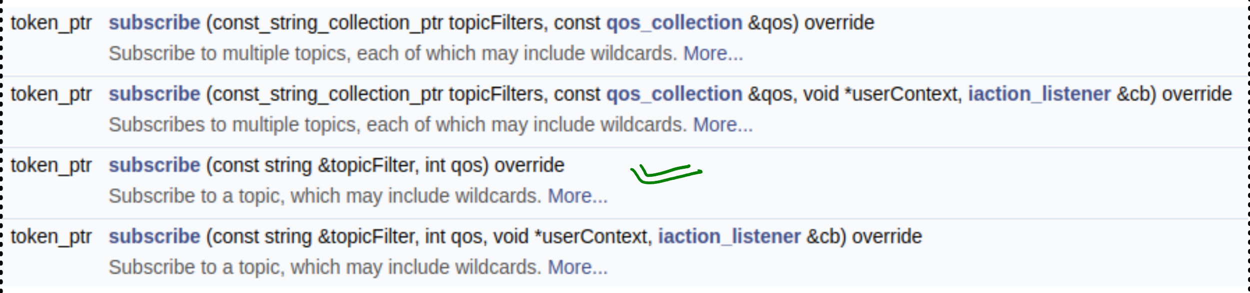
token_ptr **connect** () override
Connects to an MQTT server using the default options. More...

token_ptr **connect** (**connect_options** options) override ✓
Connects to an MQTT server using the provided connect options. More...



void **start_consuming** ()
Start consuming messages.

⇒ This initializes the client to receive messages through a queue that can be read synchronously.

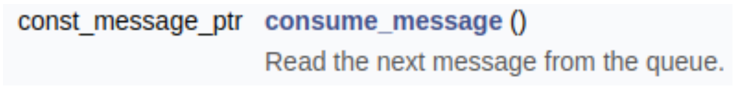


token_ptr **subscribe** (const_string_collection_ptr topicFilters, const **qos_collection** &qos) override
Subscribe to multiple topics, each of which may include wildcards. More...

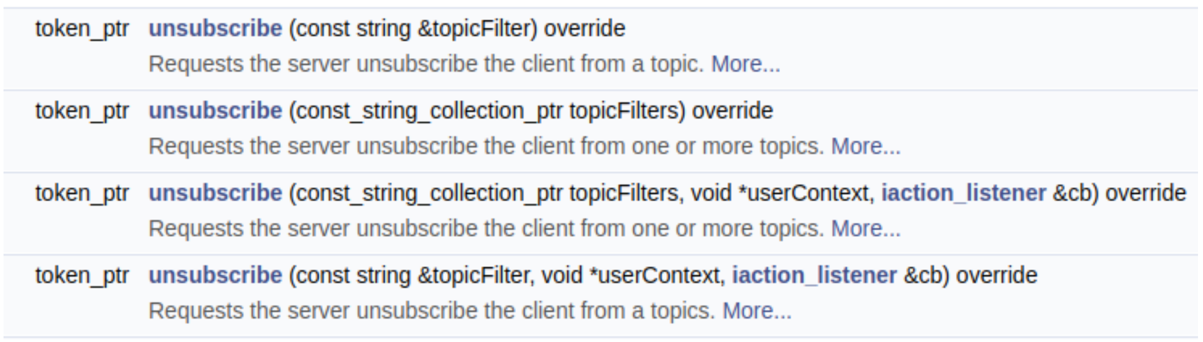
token_ptr **subscribe** (const_string_collection_ptr topicFilters, const **qos_collection** &qos, void *userContext, **iaction_listener** &cb) override
Subscribes to multiple topics, each of which may include wildcards. More...

token_ptr **subscribe** (const string &topicFilter, int qos) override ✓
Subscribe to a topic, which may include wildcards. More...

token_ptr **subscribe** (const string &topicFilter, int qos, void *userContext, **iaction_listener** &cb) override
Subscribe to a topic, which may include wildcards. More...



const_message_ptr **consume_message** ()
Read the next message from the queue.

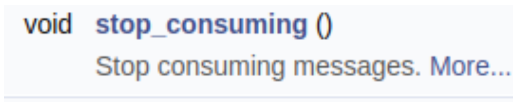


token_ptr **unsubscribe** (const string &topicFilter) override
Requests the server unsubscribe the client from a topic. More...

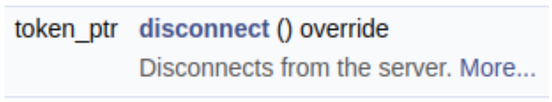
token_ptr **unsubscribe** (const_string_collection_ptr topicFilters) override
Requests the server unsubscribe the client from one or more topics. More...

token_ptr **unsubscribe** (const_string_collection_ptr topicFilters, void *userContext, **iaction_listener** &cb) override
Requests the server unsubscribe the client from one or more topics. More...

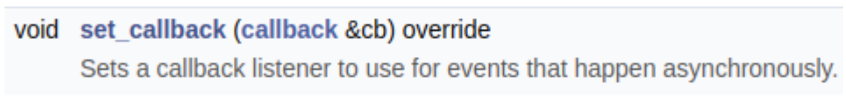
token_ptr **unsubscribe** (const string &topicFilter, void *userContext, **iaction_listener** &cb) override
Requests the server unsubscribe the client from a topics. More...



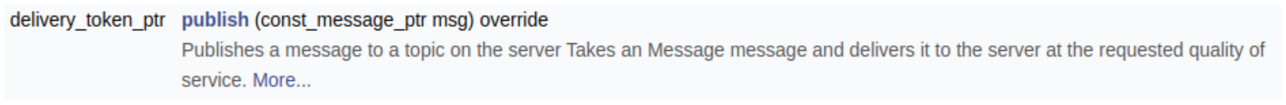
void **stop_consuming** ()
Stop consuming messages. More...

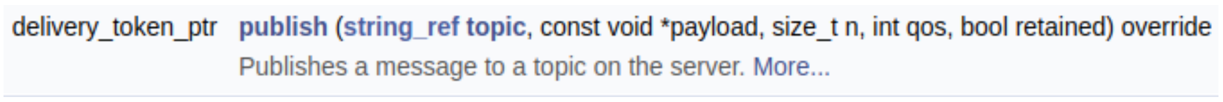


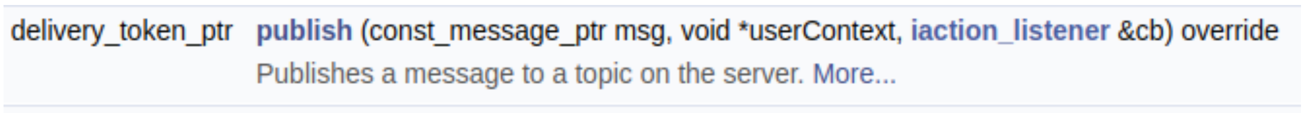
token_ptr **disconnect** () override
Disconnects from the server. More...

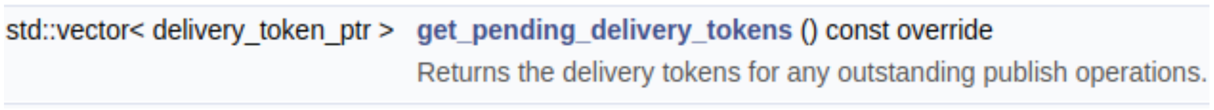


void **set_callback** (**callback** &cb) override
Sets a callback listener to use for events that happen asynchronously.


delivery_token_ptr **publish** (const_message_ptr msg) override
Publishes a message to a topic on the server Takes an Message message and delivers it to the server at the requested quality of service. More...

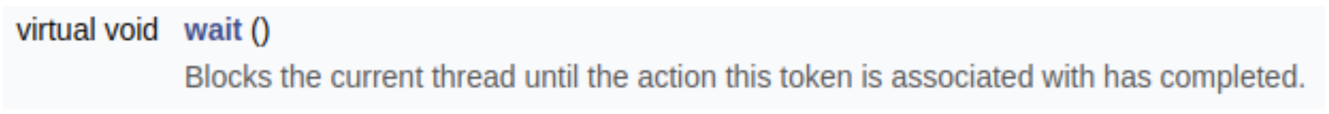

delivery_token_ptr **publish** (**string_ref topic**, const void *payload, size_t n, int qos, bool retained) override
Publishes a message to a topic on the server. More...

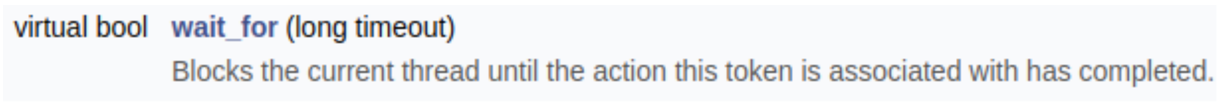

delivery_token_ptr **publish** (const_message_ptr msg, void *userContext, **iaction_listener** &cb) override
Publishes a message to a topic on the server. More...

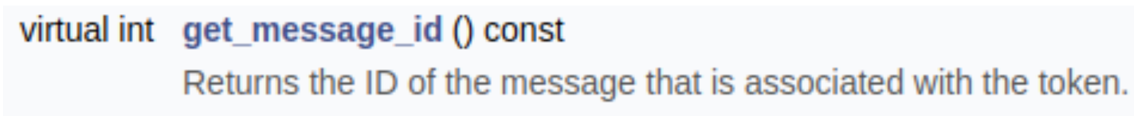

std::vector< delivery_token_ptr > **get_pending_delivery_tokens** () const override
Returns the delivery tokens for any outstanding publish operations.

3 mqtt::token

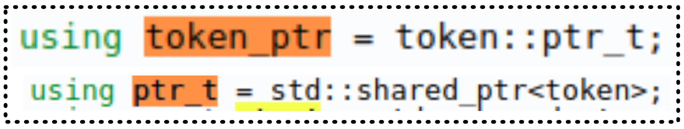
★ Public member functions

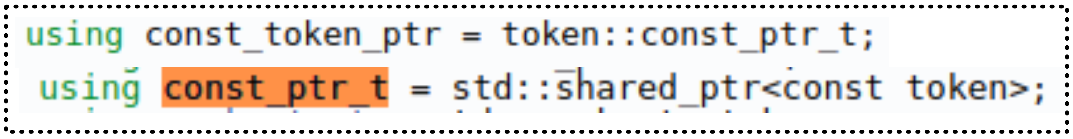

virtual void **wait** ()
Blocks the current thread until the action this token is associated with has completed.


virtual bool **wait_for** (long timeout)
Blocks the current thread until the action this token is associated with has completed.


virtual int **get_message_id** () const
Returns the ID of the message that is associated with the token.

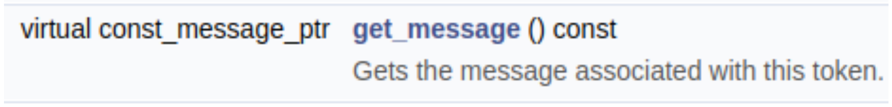
★ Type defs

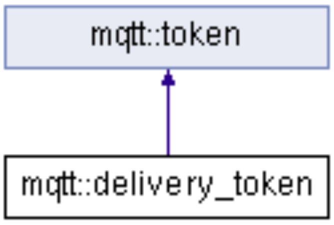

```
using token_ptr = token::ptr_t;
using ptr_t = std::shared_ptr<token>;
```


```
using const_token_ptr = token::const_ptr_t;
using const_ptr_t = std::shared_ptr<const token>;
```

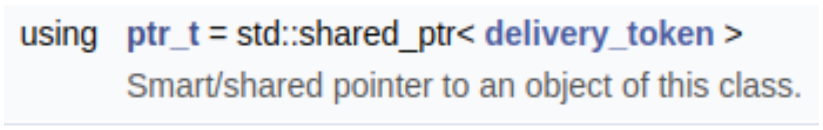
④ mqtt::delivery_token

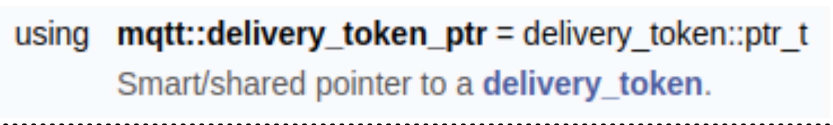
★ Member function

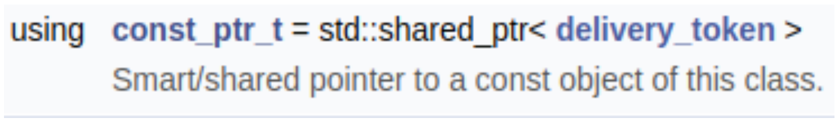

virtual const_message_ptr **get_message** () const
Gets the message associated with this token.

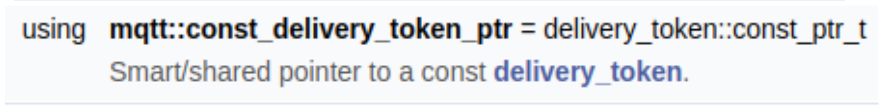


★ Typedefs


using **ptr_t** = std::shared_ptr< **delivery_token** >
Smart/shared pointer to an object of this class.

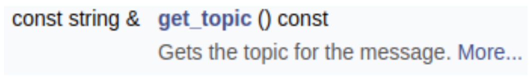

using **mqtt::delivery_token_ptr** = delivery_token::ptr_t
Smart/shared pointer to a **delivery_token**.

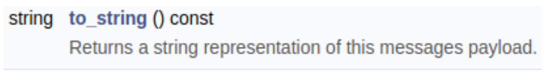

using **const_ptr_t** = std::shared_ptr< **delivery_token** >
Smart/shared pointer to a const object of this class.


using **mqtt::const_delivery_token_ptr** = delivery_token::const_ptr_t
Smart/shared pointer to a const **delivery_token**.

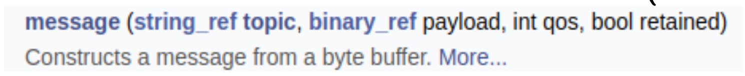
⑤ mqtt::message

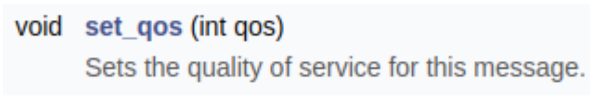
★ Public member function

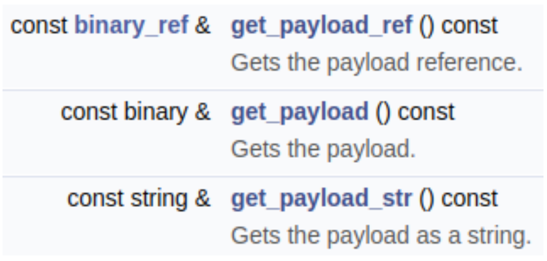

const string & **get_topic** () const
Gets the topic for the message. More...


string **to_string** () const
Returns a string representation of this messages payload.

{Whether the message should be retained by the broker.}


message (**string_ref topic**, **binary_ref** payload, int qos, bool retained)
Constructs a message from a byte buffer. More...

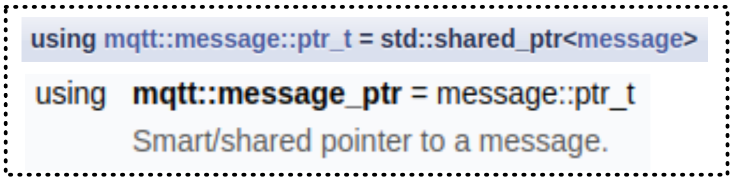

void **set_qos** (int qos)
Sets the quality of service for this message.


const **binary_ref** & **get_payload_ref** () const
Gets the payload reference.

const binary & **get_payload** () const
Gets the payload.

const string & **get_payload_str** () const
Gets the payload as a string.

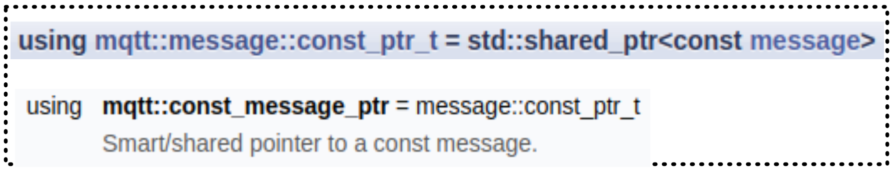
★ Type defs



```
using mqtt::message::ptr_t = std::shared_ptr<message>
```

```
using mqtt::message_ptr = message::ptr_t
```

Smart/shared pointer to a message.



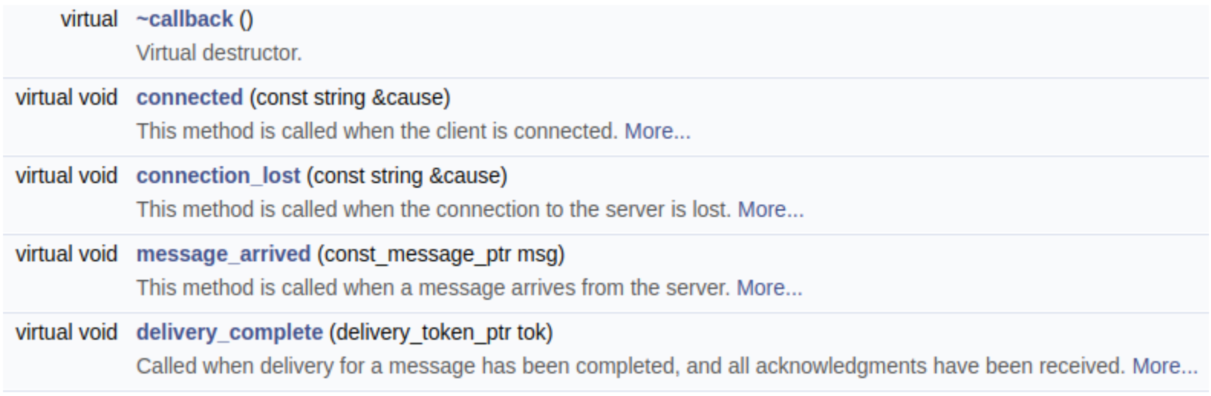
```
using mqtt::message::const_ptr_t = std::shared_ptr<const message>
```

```
using mqtt::const_message_ptr = message::const_ptr_t
```

Smart/shared pointer to a const message.

6 mqtt::callback

★ Public member functions

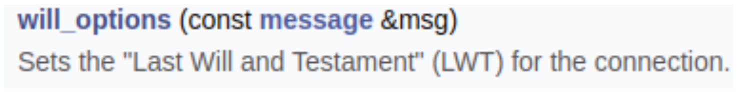


virtual	~callback () Virtual destructor.
virtual void	connected (const string &cause) This method is called when the client is connected. More...
virtual void	connection_lost (const string &cause) This method is called when the connection to the server is lost. More...
virtual void	message_arrived (const_message_ptr msg) This method is called when a message arrives from the server. More...
virtual void	delivery_complete (delivery_token_ptr tok) Called when delivery for a message has been completed, and all acknowledgments have been received. More...

7 mqtt::will_options

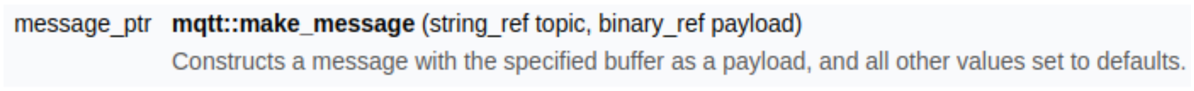
→ Holds the set of options that govern the Last Will and Testament feature.

★ Public member functions



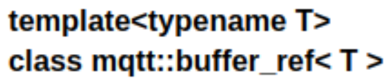
will_options (const message &msg)
Sets the "Last Will and Testament" (LWT) for the connection.

8 mqtt::make_message (function)



message_ptr	**mqtt::make_message** (string_ref topic, binary_ref payload) Constructs a message with the specified buffer as a payload, and all other values set to defaults.

9 mqtt::buffer_ref< T >



template<typename T>
class mqtt::buffer_ref< T >

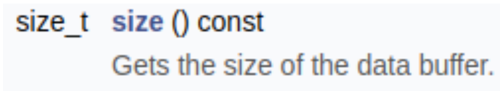
→ Each object of this class contains a reference-counted pointer to an immutable data buffer.

→ Objects can be copied freely and easily, even across threads, since all instances promise not to modify the contents of the buffer.

→ It can be reassigned to point to a different buffer.

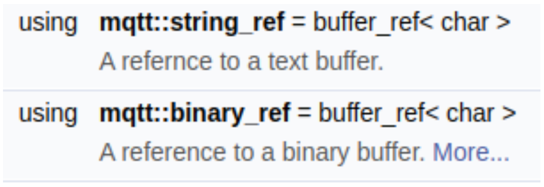
→ If no value has been assigned to a reference, then it is in a default "null" state.

★ Member function



size_t	size () const Gets the size of the data buffer.

★ Typedefs

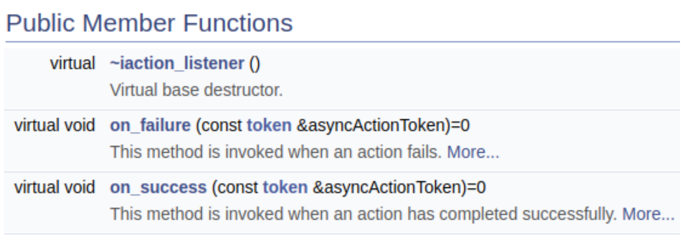


using	**mqtt::string_ref** = buffer_ref< char > A refernce to a text buffer.
using	**mqtt::binary_ref** = buffer_ref< char > A reference to a binary buffer. More...

10 mqtt::iaction_listener

⇛ Provides a mechanism for tracking the completion of an asynchronous action.

⇛ A listener is registered on a token and that token is associated with an action like connect or publish.



Public Member Functions

virtual	**~iaction_listener** () Virtual base destructor.
virtual void	**on_failure** (const **token** &asyncActionToken)=0 This method is invoked when an action fails. More...
virtual void	**on_success** (const **token** &asyncActionToken)=0 This method is invoked when an action has completed successfully. More...