MQTTClient

Generated by Doxygen 1.8.13

Contents

Chapter 1

MQTT-Client-Framework

an Objective-C native MQTT Framework http://mqtt.org

Tested with a long list of brokers

- mosquitto
- paho
- rabbitmq
- hivemq
- rsmb
- mosca
- vernemq
- emqtt
- moquette
- ActiveMQ
- Apollo
- CloudMQTT
- aws
- hbmqtt (MQTTv311 only, limitations)
- aedes

As a CocoaPod

Use the CocoaPod MQTTClient!

Add this to your Podfile:

```
pod 'MQTTClient'
```

which is a short for

```
pod 'MQTTClient/Min'
pod 'MQTTClient/Manager'
```

The Manager subspec includes the MQTTSessionManager class.

Additionally add this subspec if you want to use MQTT over Websockets:

```
pod 'MQTTClient/Websocket'
```

If you want to do your logging with CocoaLumberjack (my suggestion), use

```
pod 'MQTTClient/MinL'
pod 'MQTTClient/ManagerL'
pod 'MQTTClient/WebsocketL'
```

instead.

As a dynamic library

Or use the dynamic library created in the MQTTFramework target.

As source

Or include the source from here.

With Carthage

`Carthage`

Usage

Create a new client and connect to a broker:

```
#import "MQTTClient.h"

\@interface MyDelegate : ... <MQTTSessionDelegate>
...

        MQTTCFSocketTransport *transport = [[MQTTCFSocketTransport alloc] init];
        transport.host = @"localhost";
        transport.port = 1883;

        MQTTSession *session = [[MQTTSession alloc] init];
        session.transport = transport;

    session.delegate = self;

    [session connectAndWaitTimeout:30];  //this is part of the synchronous API
```

Subscribe to a topic:

```
[session subscribeToTopic:@"example/#" atLevel:2 subscribeHandler:^(NSError *error, NSArray<NSNumber *>
       *gQoss){
    if (error) {
        NSLog(@"Subscription failed %@", error.localizedDescription);
    } else {
        NSLog(@"Subscription sucessfull! Granted Qos: %@", gQoss);
    }
 }]; // this is part of the block API
```

Add the following to receive messages for the subscribed topics

```
 - (void)newMessage:(MQTTSession *)session
    data:(NSData *)data
    onTopic:(NSString *)topic
    qos:(MQTTQosLevel)qos
    retained:(BOOL)retained
    mid:(unsigned int)mid {
    // this is one of the delegate callbacks
}
```

Publish a message to a topic:

```
[session publishAndWaitData:data
                    onTopic:@"topic"
                     retain:NO
                    qos:MQTTQosLevelAtLeastOnce]; // this is part of the asynchronous API
```

docs

Documentation generated with doxygen `http://doxygen.org` in the `./MQTTClient/dist/documentation` subdirectory.

You may open the HTML version of the documentation here '‘./MQTTClient/dist/documentation/html/index.html'

Run `make install` in the `./MQTTClient/dist/documentation/html` subdirectory to install the the documentation as a DOCSET on your Mac.

Comparison MQTT Clients for iOS (incomplete)

Wrapper	—	-—	MQTTKit	Marquette	Moscapsule	Musqueteer	MQTT-Client	MqttSDK	CocoaMQTT
			Obj-C	Obj-C	Swift	Obj-C	Obj-C	Obj-C	Swift
Library	IBM	Paho	Mosquitto	Mosquitto	Mosquitto	Mosquitto	native	native	native

Chapter 2

MQTT-Client-Framework iOS/OSX/tvOS Release Notes

MQTT-Client-Framework 0.9.6

Release date 2017-07-25

```
[NEW] Strict parameter checking
[NEW] MQTT 3.1.1 CONNECT package does not conform #268
```

MQTT-Client-Framework 0.9.5

Release date 2017-07-07

```
[NEW] MQTTSession and MQTTTransport extension #337
```

MQTT-Client-Framework 0.9.4

Release date 2017-07-07

```
[NEW] Externally define DDLogLevel #330
```

MQTT-Client-Framework 0.9.3

Release date 2017-07-07

```
[NEW] Use xcconfig instead of compiler flag #328
```

MQTT-Client-Framework 0.9.2

Release date 2017-05-24

```
[FIX] Regression Error: MQTTSessionManager can’t reconnect after applicationDidBecomeActive #312
```

MQTT-Client-Framework 0.9.1

Release date 2017-05-24

```
[NEW] v5 adapted error handling
[FIX] Fixed the PUBACK message sent by the client having the message id twice in the message payload #317
[NEW] v5 live cycle
[NEW] Add a configurable dupTimeout property to MQTTSession #315
```

MQTT-Client-Framework 0.9.0

Release date 2017-05-10

[FIX] Fix random crashes on core data persistence #314 [FIX] use_frameworks! [FIX] Swift Tests output [FIX] CONNACK return codes [NEW] access publish data back messageDelivered is called? closes #296 [FIX] XCode 8.3.1 warnings and documentation [NEW] MQTT v5 properties [FIX] Reset PUBLISH/PUBREL command's deadline interval when connection closed #302 [NEW] initial version 5

MQTT-Client-Framework 0.8.8

Release date 2017-04-03

[FIX] Connection Retry after Closed-by-Broker Errors #297 [NEW] Configurable maxConnectionRetryInterval for MQTTSessionMananger #297 [FIX] Don't publish QoS 1 or 2 messages immediately if queued messages exists #295

MQTT-Client-Framework 0.8.7a

Release date ?

[NEW] Framework targest for macOS and tvOS [FIX] when i use TLS ,get CFNetwork SSLHandshake failed (-9807) #277

MQTT-Client-Framework 0.8.6/7

Release date 2017-01-04

[NEW] Support voip applications #243 [NEW] Add public emqtt broker to test suite [NEW] Use signals for synchronouse calls #250 [NEW] Configurable connect-in-foreground behaviour #234

[FIX] Documentation update #252 [FIX] Backward compatibility issue #253 [FIX] Publish messages by message↩ Id ascending order when using MQTTInMemoryPersistence #247 [FIX] Adds connectInForeground configuration parameter #223 [FIX] Correct crashing issue caused by locking on a object which is replaced inside the lock #220 [FIX] Use an NSLock instead of locking on an object that is often replaced [FIX] Adding MQTTSessionManager.h to the umbrella header #213 [FIX] sharing the scheme to make the project carthage compatible #198

MQTT-Client-Framework 0.8.5

Release date 2016-09-29

[FIX] CocoaLumberjack dependency resolved see #199 and README.md

MQTT-Client-Framework 0.8.4

Release date 2016-09-??

[FIX] MQTTSessionManager lastErrorCode set too late? #203

MQTT-Client-Framework 0.8.3

Release date 2016-09-23

[FIX] Cannot build after CocoaLumberjack new release #199 [FIX] Xcode8 / Swift3 compatibility

MQTT-Client-Framework 0.8.1

Release date 2016-08-10

[FIX] MQTTClient.h in podspec

MQTT-Client-Framework 0.8.0

Release date 2016-08-08

[FIX] Application extensions is not supported closes #188 [FIX] Update MQTTCoreDataPersistence.m pull request #174

MQTT-Client-Framework 0.7.9

Release date 2016-06-21

[FIX] Legacy connect method does not honor Client Certificates with default transport #160 [FIX] CFNetwork SS↩
LHandshake failed (-9807) #149

MQTT-Client-Framework 0.7.8

Release date 2016-05-23

[FIX] Fix unread and unused variables pull reques #143 [FIX] Call connect handler when connection is closed by broker without sending a CONNACK and consistent error reporting pull request #142 [NEW] Adding method for MQTTSessionManager to include protocolLevel variable pull request #140 [FIX] Fixes an issue where calling open twice on MQTTCFSocketTransport crashes pull request #131 [NEW] Add Swift test project to check #119

MQTT-Client-Framework 0.7.4

Release date 2016-03-17

[NEW] include Websockets for tvOS closes #123

MQTT-Client-Framework 0.7.3

Release date 2016-03-15

[FIX] Synchronous API timeout closes #121 [FIX] Random crash subscribing to topics closes #113

MQTT-Client-Framework 0.7.2

Release date 2016-03-03

[REVERT] Persistent store not saved to disk closes #117

MQTT-Client-Framework 0.7.0/1

Release date 2016-03-02

[FIX] Persistent store not saved to disk closes #117

MQTT-Client-Framework 0.6.8/9

Release date 2016-02-11

[FIX] Client-side certificate validations issues closes #96

MQTT-Client-Framework 0.6.7

Release date 2016-02-10

[FIX] Logs and CocoaLumberjack dependency closes #107

MQTT-Client-Framework 0.6.6

Release date 2016-02-05

[FIX] MQTTCoreDataPersistence is crashing closes #104 closes #105 [FIX] CoreData: warning: Unable to load class named 'MQTTFlow' closes #102

MQTT-Client-Framework 0.6.5

Release date 2016-01-21

[FIX] turn off verbose logging by default closes #97 [FIX] MQTTFramework.h includes all necessary files now #62

MQTT-Client-Framework 0.6.4

Release date 2016-01-17

[FIX] incorrect length checking for SUBACK #95 [FIX] incorrect length checking for UNSUBSCRIBE

MQTT-Client-Framework 0.6.3

Release date 2016-01-17

[FIX] Ignore incoming non-UTF8 topic string closes #94 [FIX] Crash b/c input stream not closed in timeout situation closes #93

MQTT-Client-Framework 0.6.2

Release date 2016-01-05

[FIX] MQTTDecoder runLoop no longer configurable closes #87 [FIX] other smaller bugs

MQTT-Client-Framework 0.6.1

Release date 2015-12-31

[FIX] CocoaPods packaging

MQTT-Client-Framework 0.6.0

Release date 2015-12-31

[NEW] refactor / cleanup test packages [NEW] abstraction protocol for persistence closes #74 [NEW] removed .framework in favor of static Xcode library [FIX] check status of websocket connection before sending [NEW] unit tested websockets [NEW] websocket transport closes #62 [NEW] refactor transport layer [NEW] Split MQTTSession.h/m for better handling closes #80 [NEW] add timeout to ...AndWait methods closes #70

[known bugs] Websockets not for MQTTSessionmanager (yet)

MQTT-Client-Framework 0.5.3

>Release date: 2015-12-02

Enhancements

[NEW] add timeout to ...AndWait methods closes #70

MQTT-Client-Framework 0.5.2

>Release date: 2015-11-28

Added dynamic framework to integrate in Swift libraries

[NEW] MQTTFramework targe added closes #78

MQTT-Client-Framework 0.5.1

>Release date: 2015-11-18

SessionManager with subscriptions feedback

[NEW] feedback on effective subscription in MQTTSessionManager closes #65

MQTT-Client-Framework 0.5.0

>Release date: 2015-11-15

API with blocks

[NEW] API with blocks. closes #68 [FIX] Messages queued while off-line are sent after 20 sec only. closes #67

MQTT-Client-Framework 0.4.0

>Release date: 2015-11-09

Multi Threading support

[FIX] Other crash issue when I publish lots of messages (multithreaded publisher). #64

MQTT-Client-Framework 0.3.7

>Release date: 2015-11-07

[FIX] wrong target OS preprocessor directives closes #63

MQTT-Client-Framework 0.3.6

>Release date: 2015-11-06

[FIX] crashes when publishing from different threads closes #61 [PROBABLE FIX] crashes when publishing from different threads #59 #56 #53 #45

MQTT-Client-Framework 0.3.5

>Release date: 2015-11-04

[NEW] Add testcases for 3.1.2-11 .. 13 (Will flags in connect message)

MQTT-Client-Framework 0.3.4

>Release date: 2015-10-28

[NEW] extensive flow tests [FIX] serialization of delegate newMessage∗ method calls [FIX] missing msgID for QoS=1 in newMessageWithFeedback

MQTT-Client-Framework 0.3.3

>Release date: 2015-10-10

[NEW] including tvOS with Cocoapods 0.39 [FIX] test coverage for topics containing 0x0000

MQTT-Client-Framework 0.3.1/2

>Release date: 2015-10-08

[NEW] comment out tvOS until Cocoapods supports it [NEW] inbound throttling closes #54

MQTT-Client-Framework 0.3.0

>Release date: 2015-10-03

[NEW] provide support for tvOS, OSX and iOS closes #50 [NEW] add messageDelivered delegate message in MQTTSessionManager closes #49 [FIX] clarification of changing subscriptions in MQTTSessionManager closes #47

MQTT-Client-Framework 0.2.6

>Release date: 2015-08-25

[NEW] MQTTSessionManager init with Persistence settings [NEW] MQTTSessionManager with optional SSL security policy

MQTT-Client-Framework 0.2.5

>Release date: 2015-08-22

[NEW] Will option on SessionManager closes #44 [NEW] Change SessionManager subscriptions while connected [FIX] Correct SessionManager subscriptions according to server session present

[NEW] zero message id is accepted on incoming publish closes #42

MQTT-Client-Framework 0.2.4

>Release date: 2015-08-16

Relaxed check for incoming Publishes (mosca 0.31.1 incompability)

[NEW] zero message id is accepted on incoming publish closes #42

MQTT-Client-Framework 0.2.3

>Release date: 2015-07-23

Important Bug Fix

[FIX] File Persistence is not saved to disk closes #41

MQTT-Client-Framework 0.2.2

>Release date: 2015-07-05

Support TLS Client Certificates

[NEW] Client Certificates

MQTT-Client-Framework 0.2.1

>Release date: 2015-06-19

Multithreading Violation with NSManagedObjectContext

[NEW] merged PR #37 - thanks [NEW] elaborated on test cases

MQTT-Client-Framework 0.2.0

>Release date: 2015-06-03

Add SSL Certificates Pinning and Self-Signed Certificates support

[NEW] merge PR #34

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

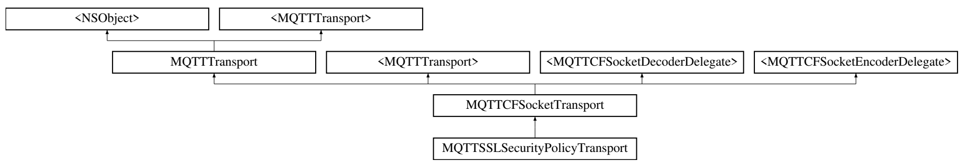
Chapter 5

Class Documentation

5.1 MQTTCFSocketTransport Class Reference

```
#import <MQTTCFSocketTransport.h>
```

Inheritance diagram for MQTTCFSocketTransport:


<NSObject>
<MQTTTransport>
MQTTTransport
<MQTTTransport>
<MQTTCFSocketDecoderDelegate>
<MQTTCFSocketEncoderDelegate>
MQTTCFSocketTransport
MQTTSSLSecurityPolicyTransport

Class Methods

- (NSArray ∗) + clientCertsFromP12:passphrase:

Properties

- NSString ∗ host
- UInt32 port
- BOOL tls
- BOOL voip
- NSArray ∗ certificates

5.1.1 Detailed Description

MQTTCFSocketTransport implements an MQTTTransport on top of CFNetwork

5.1.2 Method Documentation

5.1.2.1 clientCertsFromP12:passphrase:()

```
+ (NSArray ∗) clientCertsFromP12:
              (NSString ∗) path
             passphrase:(NSString ∗) passphrase
```

reads the content of a PKCS12 file and converts it to an certificates array for initWith...

Parameters

path	the path to a PKCS12 file
passphrase	the passphrase to unlock the PKCS12 file

Returns

a certificates array or nil if an error occured

```
NSString *path = [[NSBundle bundleForClass:[MQTTClientTests class]] pathForResource:@"filename"
ofType:@"p12"];

NSArray *myCerts = [MQTTCFSocketTransport clientCertsFromP12:path passphrase:@"
      passphrase"];
if (myCerts) {

self.session = [[MQTTSession alloc] init];
...
self.session.certificates = myCerts;

[self.session connect];
...
}
```

5.1.3 Property Documentation

5.1.3.1 certificates

```
- (NSArray*) certificates  [read], [write], [nonatomic], [strong]
```

certificates An identity certificate used to reply to a server requiring client certificates according to the description given for SSLSetCertificate(). You may build the certificates array yourself or use the sundry method clientCert↩ FromP12.

5.1.3.2 host

```
- (NSString*) host  [read], [write], [nonatomic], [strong]
```

host an NSString containing the hostName or IP address of the host to connect to defaults to "localhost"

5.1.3.3 port

```
- (UInt32) port  [read], [write], [nonatomic], [assign]
```

port an unsigned 32 bit integer containing the IP port number to connect to defaults to 1883

5.1.3.4 tls

```
- (BOOL) tls  [read], [write], [nonatomic], [assign]
```

tls a boolean indicating whether the transport should be using security defaults to NO

5.1.3.5 voip

```
- (BOOL) voip  [read], [write], [nonatomic], [assign]
```

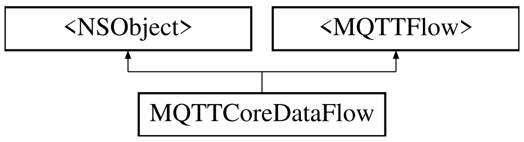
Require for VoIP background service defaults to NO

The documentation for this class was generated from the following file:

- MQTTCFSocketTransport.h

5.2 MQTTCoreDataFlow Class Reference

Inheritance diagram for MQTTCoreDataFlow:



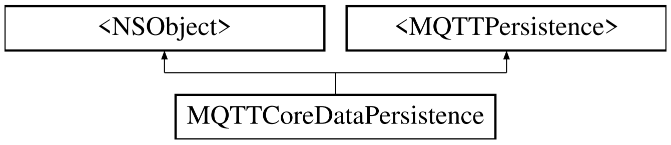
Additional Inherited Members

The documentation for this class was generated from the following file:

- MQTTCoreDataPersistence.h

5.3 MQTTCoreDataPersistence Class Reference

Inheritance diagram for MQTTCoreDataPersistence:



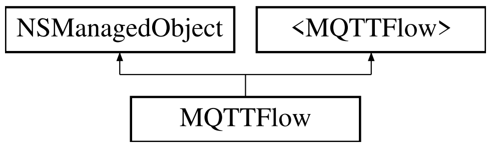
Additional Inherited Members

The documentation for this class was generated from the following file:

- MQTTCoreDataPersistence.h

5.4 MQTTFlow Class Reference

Inheritance diagram for MQTTFlow:



Additional Inherited Members

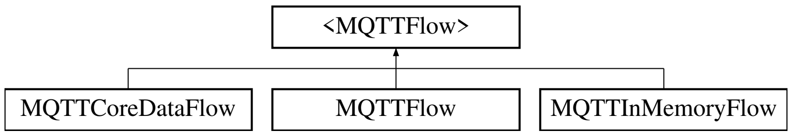
The documentation for this class was generated from the following file:

- MQTTCoreDataPersistence.h

5.5 <MQTTFlow> Protocol Reference

```
#import <MQTTPersistence.h>
```

Inheritance diagram for <MQTTFlow>:



Properties

- NSString ∗ clientId
- NSNumber ∗ incomingFlag
- NSNumber ∗ retainedFlag
- NSNumber ∗ commandType
- NSNumber ∗ qosLevel
- NSNumber ∗ messageId
- NSString ∗ topic
- NSData ∗ data
- NSDate ∗ deadline

5.5.1 Detailed Description

MQTTFlow is an abstraction of the entity to be stored for persistence

5.5.2 Property Documentation

5.5.2.1 clientId

```
- (NSString*) clientId  [read], [write], [nonatomic], [strong]
```

The clientID of the flow element

5.5.2.2 commandType

```
- (NSNumber*) commandType  [read], [write], [nonatomic], [strong]
```

The MQTTCommandType of the flow element, might be MQTT_None for offline queueing

5.5.2.3 data

```
- (NSData*) data  [read], [write], [nonatomic], [strong]
```

The data of the flow element

5.5.2.4 deadline

```
- (NSDate*) deadline  [read], [write], [nonatomic], [strong]
```

The deadline of the flow elelment before (re)trying transmission

5.5.2.5 incomingFlag

```
- (NSNumber*) incomingFlag  [read], [write], [nonatomic], [strong]
```

The flag indicating incoming or outgoing flow element

5.5.2.6 messageId

```
- (NSNumber*) messageId  [read], [write], [nonatomic], [strong]
```

The messageId of the flow element

5.5.2.7 qosLevel

```
- (NSNumber*) qosLevel  [read], [write], [nonatomic], [strong]
```

The MQTTQosLevel of the flow element

5.5.2.8 retainedFlag

```
- (NSNumber*) retainedFlag  [read], [write], [nonatomic], [strong]
```

The flag indicating if the flow element is retained

5.5.2.9 topic

```
- (NSString*) topic  [read], [write], [nonatomic], [strong]
```

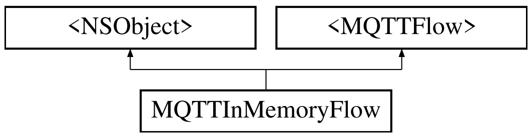
The topic of the flow element

The documentation for this protocol was generated from the following file:

- MQTTPersistence.h

5.6 MQTTInMemoryFlow Class Reference

Inheritance diagram for MQTTInMemoryFlow:



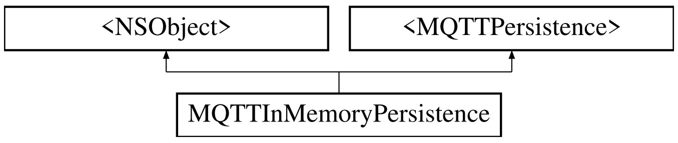
Additional Inherited Members

The documentation for this class was generated from the following file:

- MQTTInMemoryPersistence.h

5.7 MQTTInMemoryPersistence Class Reference

Inheritance diagram for MQTTInMemoryPersistence:



Additional Inherited Members

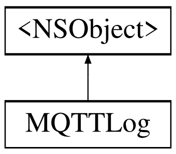
The documentation for this class was generated from the following file:

- MQTTInMemoryPersistence.h

5.8 MQTTLog Class Reference

```
#import <MQTTLog.h>
```

Inheritance diagram for MQTTLog:



Class Methods

- (void) + setLogLevel:

5.8.1 Detailed Description

MQTTLog lets you define the log level for MQTTClient independently of using CocoaLumberjack

5.8.2 Method Documentation

5.8.2.1 setLogLevel:()

```
+ (void) setLogLevel:
            (DDLogLevel) logLevel
```

setLogLevel controls the log level for MQTTClient

Parameters

logLevel	as follows:

default for DEBUG builds is DDLogLevelVerbose default for RELEAE builds is DDLogLevelWarning

Available log levels: DDLogLevelAll DDLogLevelVerbose DDLogLevelDebug DDLogLevelInfo DDLogLevelWarning DDLogLevelError DDLogLevelOff

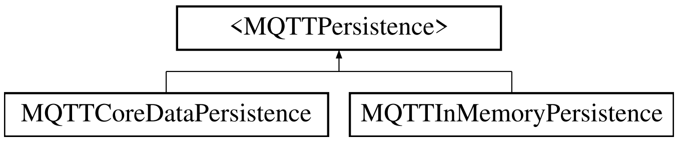
The documentation for this class was generated from the following file:

- MQTTLog.h

5.9 <MQTTPersistence> Protocol Reference

```
#import <MQTTPersistence.h>
```

Inheritance diagram for <MQTTPersistence>:



Instance Methods

- (NSUInteger) - windowSize:
- (id< MQTTFlow >) - storeMessageForClientId:topic:data:retainFlag:qos:msgId:incomingFlag:command↩ Type:deadline:
- (void) - deleteFlow:
- (void) - deleteAllFlowsForClientId:
- (NSArray ∗) - allFlowsforClientId:incomingFlag:
- (id< MQTTFlow >) - flowforClientId:incomingFlag:messageId:
- (void) - sync

Properties

- NSUInteger maxWindowSize
- NSUInteger maxMessages
- BOOL persistent
- NSUInteger maxSize

5.9.1 Detailed Description

The MQTTPersistence protocol is an abstraction of persistence classes for MQTTSession

5.9.2 Method Documentation

5.9.2.1 allFlowsforClientId:incomingFlag:()

```
- (NSArray *) allFlowsforClientId:
              (NSString *) clientId
              incomingFlag:(BOOL) incomingFlag
```

Retrieves all MQTTFlow elements of a clientId and direction

Parameters

clientId	whos MQTTFlows should be retrieved
incomingFlag	specifies the wether incoming or outgoing flows should be retrieved

Returns

an NSArray of the retrieved MQTTFlow elements

5.9.2.2 deleteAllFlowsForClientId:()

```
- (void) deleteAllFlowsForClientId:
            (NSString *) clientId
```

Deletes all MQTTFlow elements of a clientId

Parameters

client↩ Id	the client Id identifying all MQTTFlows to be deleted

5.9.2.3 deleteFlow:()

```
- (void) deleteFlow:
            (id< MQTTFlow >) flow
```

Deletes an MQTTFlow element

Parameters

flow	the MQTTFlow to delete

5.9.2.4 flowforClientId:incomingFlag:messageId:()

```
- (id<MQTTFlow>) flowforClientId:
            (NSString *) clientId
            incomingFlag:(BOOL) incomingFlag
            messageId:(UInt16) messageId
```

Retrieves an MQTTFlow element

Parameters

clientId	to which the MQTTFlow belongs to
incomingFlag	specifies the direction of the flow
messageId	specifies the message Id of the flow

Returns

the retrieved MQTTFlow element or nil if the elememt was not found

5.9.2.5 storeMessageForClientId:topic:data:retainFlag:qos:msgId:incomingFlag:commandType:deadline:()

```
- (id<MQTTFlow>) storeMessageForClientId:
              (NSString *) clientId
             topic:(NSString *) topic
             data:(NSData *) data
             retainFlag:(BOOL) retainFlag
             qos:(MQTTQosLevel) qos
             msgId:(UInt16) msgId
             incomingFlag:(BOOL) incomingFlag
             commandType:(UInt8) commandType
             deadline:(NSDate *) deadline
```

Stores one new message

Parameters

clientId	identifying the session
topic	the topic of the message
data	the message's data
retainFlag	the retain flag of the message
qos	the quality of service of the message
msgId	the id of the message or zero for qos zero
incomingFlag	the direction of the message
commandType	the command of the message
deadline	the deadline of the message for repetitions

Returns

the created MQTTFlow element or nil if the maxWindowSize has been exceeded

5.9.2.6 sync()

```
- (void) sync
```

sync is called to allow the MQTTPersistence implemetation to save data permanently

5.9.2.7 windowSize:()

```
- (NSUInteger) windowSize:
            (NSString *) clientId
```

The current Window Size for outgoing inflight messages per clientID.

Parameters

client↵ Id	identifying the session

Returns

the current size of the outgoing inflight window

5.9.3 Property Documentation

5.9.3.1 maxMessages

```
- (NSUInteger) maxMessages  [read], [write], [nonatomic], [assign]
```

The maximum number of messages kept per clientID and direction. Defaults to 1024

5.9.3.2 maxSize

```
- (NSUInteger) maxSize  [read], [write], [nonatomic], [assign]
```

The maximum size of the storage used for persistence in total in bytes. Defaults to 1024∗1024 bytes

5.9.3.3 maxWindowSize

```
- (NSUInteger) maxWindowSize  [read], [write], [nonatomic], [assign]
```

The maximum Window Size for outgoing inflight messages per clientID. Defaults to 16

5.9.3.4 persistent

```
- (BOOL) persistent  [read], [write], [nonatomic], [assign]
```

Indicates if the persistence implementation should make the information permannent. Defaults to NO

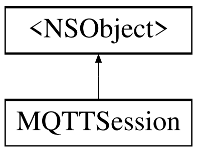
The documentation for this protocol was generated from the following file:

- MQTTPersistence.h

5.10 MQTTSession Class Reference

```
#import <MQTTSession.h>
```

Inheritance diagram for MQTTSession:



Instance Methods

- (void) - connect
- (void) - connectWithConnectHandler:
- (void) - disconnect
- (void) - disconnectWithReturnCode:sessionExpiryInterval:reasonString:userProperty:
- (MQTTSession ∗) - init
- (UInt16) - subscribeToTopic:atLevel:
- (UInt16) - subscribeToTopic:atLevel:subscribeHandler:
- (UInt16) - subscribeToTopics:
- (UInt16) - subscribeToTopics:subscribeHandler:
- (UInt16) - unsubscribeTopic:
- (UInt16) - unsubscribeTopic:unsubscribeHandler:
- (UInt16) - unsubscribeTopics:
- (UInt16) - unsubscribeTopics:unsubscribeHandler:
- (UInt16) - publishData:onTopic:retain:qos:
- (UInt16) - publishData:onTopic:retain:qos:publishHandler:
- (void) - closeWithDisconnectHandler:
- (void) - closeWithReturnCode:sessionExpiryInterval:reasonString:userProperty:disconnectHandler:
- (void) - close

Properties

- id< MQTTSessionDelegate > delegate
- id< MQTTPersistence > persistence
- MQTTConnectHandler connectHandler
- void(^ connectionHandler)(MQTTSessionEvent event)
- void(^ messageHandler)(NSData ∗message, NSString ∗topic)
- MQTTSessionStatus status
- BOOL sessionPresent
- NSString ∗ host
- UInt32 port
- NSString ∗ clientId
- NSString ∗ userName
- NSString ∗ password
- UInt16 keepAliveInterval
- NSNumber ∗ serverKeepAlive
- UInt16 effectiveKeepAlive
- double dupTimeout
- BOOL cleanSessionFlag

- BOOL willFlag
- NSString ∗ willTopic
- NSData ∗ willMsg
- MQTTQosLevel willQoS
- BOOL willRetainFlag
- MQTTProtocolVersion protocolLevel
- NSNumber ∗ sessionExpiryInterval
- NSString ∗ authMethod
- NSData ∗ authData
- NSNumber ∗ requestProblemInformation
- NSNumber ∗ willDelayInterval
- NSNumber ∗ requestResponseInformation
- NSNumber ∗ receiveMaximum
- NSNumber ∗ topicAliasMaximum
- NSDictionary< NSString ∗, NSString ∗ > ∗ userProperty
- NSNumber ∗ maximumPacketSize
- NSRunLoop ∗ runLoop
- NSString ∗ runLoopMode
- MQTTMessage ∗ connectMessage
- id< MQTTTransport > transport
- NSArray ∗ certificates
- BOOL voip

5.10.1 Detailed Description

Session implements the MQTT protocol for your application

5.10.2 Method Documentation

5.10.2.1 close()

```
- (void) close
```

closes an MQTTSession gracefully

5.10.2.2 closeWithDisconnectHandler:()

```
- (void) closeWithDisconnectHandler:
            (MQTTDisconnectHandler) disconnectHandler
```

closes an MQTTSession gracefully

If the connection was successfully established before, a DISCONNECT is sent.

Parameters

disconnectHandler	identifies a block which is executed on successfull or unsuccessfull disconnect. Might be nil. error is nil in the case of a successful disconnect

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connect];

...

[session closeWithDisconnectHandler^(NSError *error) {
   if (error) {
       NSLog(@"Error Disconnect %@", error.localizedDescription);
   }
   NSLog(@"Session closed");
}];
```

5.10.2.3 closeWithReturnCode:sessionExpiryInterval:reasonString:userProperty:disconnectHandler:()

```
- (void) closeWithReturnCode:
              (MQTTReturnCode) returnCode
             sessionExpiryInterval:(NSNumber *) sessionExpiryInterval
             reasonString:(NSString *) reasonString
             userProperty:(NSDictionary< NSString *, NSString * > *) userProperty
             disconnectHandler:(MQTTDisconnectHandler) disconnectHandler
```

close V5

Parameters

returnCode	the returncode send to the broker
sessionExpiryInterval	the time in seconds before the session can be deleted
reasonString	a string explaining the reason
userProperty	additional dictionary of user key/value combinations
disconnectHandler	will be called when the disconnect finished

5.10.2.4 connect()

```
- (void) connect
```

connect to the given host through the given transport with the given MQTT session parameters asynchronously

5.10.2.5 connectWithConnectHandler:()

```
- (void) connectWithConnectHandler:
              (MQTTConnectHandler) connectHandler
```

connects to the specified MQTT server

Parameters

connectHandler	identifies a block which is executed on successfull or unsuccessfull connect. Might be nil error is nil in the case of a successful connect sessionPresent indicates in MQTT 3.1.1 if persistent session data was present at the server returns nothing and returns immediately. To check the connect results, register as an MQTTSessionDelegate and • watch for events • watch for connect or connectionRefused messages • watch for error messages or use the connectHandler block

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connectWithConnectHandler:^(NSError *error, BOOL sessionPresent) {
if (error) {
NSLog(@"Error Connect %@", error.localizedDescription);
} else {
NSLog(@"Connected sessionPresent:%d", sessionPresent);
}
}];
```

5.10.2.6 disconnect()

```
- (void) disconnect
```

disconnect gracefully

5.10.2.7 disconnectWithReturnCode:sessionExpiryInterval:reasonString:userProperty:()

```
- (void) disconnectWithReturnCode:
                (MQTTReturnCode) returnCode
                sessionExpiryInterval:(NSNumber *) sessionExpiryInterval
                reasonString:(NSString *) reasonString
                userProperty:(NSDictionary< NSString *, NSString * > *) userProperty
```

disconnect V5

Parameters

returnCode	the returncode send to the broker
sessionExpiryInterval	the time in seconds before the session can be deleted
reasonString	a string explaining the reason
userProperty	additional dictionary of user key/value combinations

5.10.2.8 init()

```
- (MQTTSession *) init
```

initialises the MQTT session with default values

Returns

the initialised MQTTSession object

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
```

5.10.2.9 publishData:onTopic:retain:qos:()

```
- (UInt16) publishData:
            (NSData *) data
            onTopic:(NSString *) topic
            retain:(BOOL) retainFlag
            qos:(MQTTQosLevel) qos
```

publishes data on a given topic at a specified QoS level and retain flag

Parameters

data	the data to be sent. length may range from 0 to 268,435,455 - 4 - *lengthof-topic* bytes. Defaults to length 0.
topic	the Topic to identify the data
retainFlag	if YES, data is stored on the MQTT broker until overwritten by the next publish with retainFlag = YES
qos	specifies the Quality of Service for the publish qos can be 0, 1, or 2.

Returns

the Message Identifier of the PUBLISH message. Zero if qos 0. If qos 1 or 2, zero if message was dropped

Note

returns immediately. To check results, register as an MQTTSessionDelegate and watch for events.

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connect];

[session publishData:[@"Sample Data" dataUsingEncoding:NSUTF8StringEncoding]
topic:@"example/data"
retain:YES
qos:1];
```

5.10.2.10 publishData:onTopic:retain:qos:publishHandler:()

```
- (UInt16) publishData:
              (NSData *) data
              onTopic:(NSString *) topic
              retain:(BOOL) retainFlag
              qos:(MQTTQosLevel) qos
              publishHandler:(MQTTPublishHandler) publishHandler
```

publishes data on a given topic at a specified QoS level and retain flag

Parameters

data	the data to be sent. length may range from 0 to 268,435,455 - 4 - *lengthof-topic* bytes. Defaults to length 0.
topic	the Topic to identify the data
retainFlag	if YES, data is stored on the MQTT broker until overwritten by the next publish with retainFlag = YES
qos	specifies the Quality of Service for the publish qos can be 0, 1, or 2.
publishHandler	identifies a block which is executed on successfull or unsuccessfull publsh. Might be nil error is nil in the case of a successful connect sessionPresent indicates in MQTT 3.1.1 if persistent session data was present at the server

Returns

the Message Identifier of the PUBLISH message. Zero if qos 0. If qos 1 or 2, zero if message was dropped

Note

returns immediately. To check results, register as an MQTTSessionDelegate and watch for events.

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connect];

[session publishData:[@"Sample Data" dataUsingEncoding:NSUTF8StringEncoding]
topic:@"example/data"
retain:YES
qos:1
publishHandler:^(NSError *error){
if (error) {
DDLogVerbose(@"error: %@ %@", error.localizedDescription, payload);
} else {
DDLogVerbose(@"delivered:%@", payload);
delivered++;
}
}];
```

5.10.2.11 subscribeToTopic:atLevel:()

```
- (UInt16) subscribeToTopic:
              (NSString *) topic
              atLevel:(MQTTQosLevel) qosLevel
```

subscribes to a topic at a specific QoS level

Parameters

topic	see subscribeToTopic:atLevel:subscribeHandler: for description
qosLevel	see subscribeToTopic:atLevel:subscribeHandler: for description

Returns

the Message Identifier of the SUBSCRIBE message.

Note

returns immediately. To check results, register as an MQTTSessionDelegate and watch for events.

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connect];
...
[session subscribeToTopic:@"example/#" atLevel:2];
```

5.10.2.12 subscribeToTopic:atLevel:subscribeHandler:()

```
- (UInt16) subscribeToTopic:
              (NSString *) topic
              atLevel:(MQTTQosLevel) qosLevel
              subscribeHandler:(MQTTSubscribeHandler) subscribeHandler
```

subscribes to a topic at a specific QoS level

Parameters

topic	the Topic Filter to subscribe to.
qosLevel	specifies the QoS Level of the subscription. qosLevel can be 0, 1, or 2.
subscribeHandler	identifies a block which is executed on successfull or unsuccessfull subscription. Might be nil. error is nil in the case of a successful subscription. In this case gQoss represents an array of grantes Qos

Returns

the Message Identifier of the SUBSCRIBE message.

Note

returns immediately. To check results, register as an MQTTSessionDelegate and watch for events.

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connect];
```

```
...
[session subscribeToTopic:@"example/#" atLevel:2 subscribeHandler:^(NSError *error, NSArray<NSNumber *> *
      gQoss){
   if (error) {
       NSLog(@"Subscription failed %@", error.localizedDescription);
   } else {
       NSLog(@"Subscription sucessfull! Granted Qos: %@", gQoss);
   }
}];
```

5.10.2.13 subscribeToTopics:()

```
- (UInt16) subscribeToTopics:
               (NSDictionary< NSString *, NSNumber * > *) topics
```

subscribes a number of topics

Parameters

topics	an NSDictionary<NSString *, NSNumber *> containing the Topic Filters to subscribe to as keys and the corresponding QoS as NSNumber values

Returns

the Message Identifier of the SUBSCRIBE message.

Note

returns immediately. To check results, register as an MQTTSessionDelegate and watch for events.

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connect];

[session subscribeToTopics:@{
@"example/#": @(0),
@"example/status": @(2),
@"other/#": @(1)
}];
```

5.10.2.14 subscribeToTopics:subscribeHandler:()

```
- (UInt16) subscribeToTopics:
               (NSDictionary< NSString *, NSNumber * > *) topics
               subscribeHandler:(MQTTSubscribeHandler) subscribeHandler
```

subscribes a number of topics

Parameters

topics	an NSDictionary<NSString ∗, NSNumber ∗> containing the Topic Filters to subscribe to as keys and the corresponding QoS as NSNumber values
subscribeHandler	identifies a block which is executed on successfull or unsuccessfull subscription. Might be nil. error is nil in the case of a successful subscription. In this case gQoss represents an array of grantes Qos

Returns

the Message Identifier of the SUBSCRIBE message.

Note

returns immediately. To check results, register as an MQTTSessionDelegate and watch for events.

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connect];

[session subscribeToTopics:@{
   @"example/#": @(0),
   @"example/status": @(2),
   @"other/#": @(1)
} subscribeHandler:^(NSError *error, NSArray<NSNumber *> *gQoss){
   if (error) {
       NSLog(@"Subscription failed %@", error.localizedDescription);
   } else {
       NSLog(@"Subscription sucessfull! Granted Qos: %@", gQoss);
   }
}];
```

5.10.2.15 unsubscribeTopic:()

```
- (UInt16) unsubscribeTopic:
              (NSString *) topic
```

unsubscribes from a topic

Parameters

topic	the Topic Filter to unsubscribe from.

Returns

the Message Identifier of the UNSUBSCRIBE message.

Note

returns immediately. To check results, register as an MQTTSessionDelegate and watch for events.

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connect];

[session unsubscribeTopic:@"example/#"];
```

5.10.2.16 unsubscribeTopic:unsubscribeHandler:()

```
- (UInt16) unsubscribeTopic:
              (NSString *) topic
              unsubscribeHandler:(MQTTUnsubscribeHandler) unsubscribeHandler
```

unsubscribes from a topic

Parameters

topic	the Topic Filter to unsubscribe from.
unsubscribeHandler	identifies a block which is executed on successfull or unsuccessfull subscription. Might be nil. error is nil in the case of a successful subscription. In this case gQoss represents an array of grantes Qos

Returns

the Message Identifier of the UNSUBSCRIBE message.

Note

returns immediately.

5.10.2.17 unsubscribeTopics:()

```
- (UInt16) unsubscribeTopics:
              (NSArray< NSString * > *) topics
```

unsubscribes from a number of topics

Parameters

topics	an NSArray<NSString *> of topics to unsubscribe from

Returns

the Message Identifier of the UNSUBSCRIBE message.

Note

returns immediately. To check results, register as an MQTTSessionDelegate and watch for events.

```
#import "MQTTClient.h"

MQTTSession *session = [[MQTTSession alloc] init];
...
[session connect];

[session unsubscribeTopics:@[
@"example/#",
@"example/status",
@"other/#"
]];
```

5.10.2.18 unsubscribeTopics:unsubscribeHandler:()

```
- (UInt16) unsubscribeTopics:
              (NSArray< NSString * > *) topics
              unsubscribeHandler:(MQTTUnsubscribeHandler) unsubscribeHandler
```

unsubscribes from a number of topics

Parameters

topics	an NSArray<NSString *> of topics to unsubscribe from
unsubscribeHandler	identifies a block which is executed on successfull or unsuccessfull subscription. Might be nil. error is nil in the case of a successful subscription. In this case gQoss represents an array of grantes Qos

Returns

the Message Identifier of the UNSUBSCRIBE message.

Note

returns immediately.

5.10.3 Property Documentation

5.10.3.1 authData

```
- (NSData*) authData  [read], [write], [nonatomic], [strong]
```

authData specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.2 authMethod

```
- (NSString*) authMethod  [read], [write], [nonatomic], [strong]
```

authMethod specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.3 certificates

```
- (NSArray*) certificates  [read], [write], [nonatomic], [strong]
```

certificates an NSArray holding client certificates or nil

5.10.3.4 cleanSessionFlag

```
- (BOOL) cleanSessionFlag  [read], [write], [nonatomic], [assign]
```

leanSessionFlag specifies if the server should discard previous session information.

5.10.3.5 clientId

```
- (NSString*) clientId  [read], [write], [nonatomic], [strong]
```

The Client Identifier identifies the Client to the Server. If nil, a random clientId is generated.

5.10.3.6 connectHandler

```
- (MQTTConnectHandler) connectHandler  [read], [write], [nonatomic], [copy]
```

block called once when connection is established

5.10.3.7 connectionHandler

```
- (void(^ connectionHandler) (MQTTSessionEvent event))  [read], [write], [atomic], [strong]
```

block called when connection is established

5.10.3.8 connectMessage

```
- (MQTTMessage*) connectMessage  [read], [write], [nonatomic], [strong]
```

for mqttio-OBJC backward compatibility the connect message used is stored here

5.10.3.9 delegate

```
- (id<MQTTSessionDelegate>) delegate  [read], [write], [nonatomic], [weak]
```

set this member variable to receive delegate messages

```
#import "MQTTClient.h"

@interface MyClass : NSObject <MQTTSessionDelegate>
...
@end

...
MQTTSession *session = [[MQTTSession alloc] init];
session.delegate = self;
...
- (void)handleEvent:(MQTTSession *)session
       event:(MQTTSessionEvent)eventCode
       error:(NSError *)error {
    ...
}
- (void)newMessage:(MQTTSession *)session
       data:(NSData *)data
       onTopic:(NSString *)topic
       qos:(MQTTQosLevel)qos
       retained:(BOOL)retained
       mid:(unsigned int)mid {
    ...
}
```

5.10.3.10 dupTimeout

```
- (double) dupTimeout  [read], [write], [nonatomic], [assign]
```

dupTimeout If PUBACK or PUBREC not received, message will be resent after this interval

5.10.3.11 effectiveKeepAlive

```
- (UInt16) effectiveKeepAlive  [read], [nonatomic], [assign]
```

effectiveKeepAlive is a time interval measured in seconds It indicates the effective keep alive interval after a successfull connect where keepAliveInterval might have been overridden by the broker.

5.10.3.12 host

```
- (NSString*) host  [read], [atomic], [assign]
```

host an NSString containing the hostName or IP address of the Server

5.10.3.13 keepAliveInterval

```
- (UInt16) keepAliveInterval  [read], [write], [nonatomic], [assign]
```

see keepAliveInterval The Keep Alive is a time interval measured in seconds. The MQTTClient ensures that the interval between Control Packets being sent does not exceed the Keep Alive value. In the absence of sending any other Control Packets, the Client sends a PINGREQ Packet.

5.10.3.14 maximumPacketSize

```
- (NSNumber*) maximumPacketSize  [read], [write], [nonatomic], [strong]
```

maximumPacketSize specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.15 messageHandler

```
- (void(^ messageHandler) (NSData *message, NSString *topic))  [read], [write], [atomic], [strong]
```

block called when message is received

5.10.3.16 password

```
- (NSString*) password  [read], [write], [nonatomic], [strong]
```

see password an NSString object containing the user's password. If userName is nil, password must be nil as well.

5.10.3.17 persistence

```
- (id<MQTTPersistence>) persistence  [read], [write], [nonatomic], [strong]
```

Control MQTT persistence by setting the properties of persistence before connecting to an MQTT broker. The settings are specific to a clientId.

persistence.persistent = YES or NO (default) to establish file or in memory persistence. IMPORTANT: set immediately after creating the MQTTSession before calling any other method. Otherwise the default value (NO) will be used for this session.

persistence.maxWindowSize (a positive number, default is 16) to control the number of messages sent before waiting for acknowledgement in Qos 1 or 2. Additional messages are stored and transmitted later.

persistence.maxSize (a positive number of bytes, default is 64 MB) to limit the size of the persistence file. Messages published after the limit is reached are dropped.

persistence.maxMessages (a positive number, default is 1024) to limit the number of messages stored. Additional messages published are dropped.

Messages are deleted after they have been acknowledged.

5.10.3.18 port

```
- (UInt32) port  [read], [atomic], [assign]
```

port an unsigned 32 bit integer containing the IP port number of the Server

5.10.3.19 protocolLevel

```
- (MQTTProtocolVersion) protocolLevel  [read], [write], [nonatomic], [assign]
```

protocolLevel specifies the protocol to be used

5.10.3.20 receiveMaximum

```
- (NSNumber*) receiveMaximum  [read], [write], [nonatomic], [strong]
```

receiveMaximum specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.21 requestProblemInformation

```
- (NSNumber*) requestProblemInformation  [read], [write], [nonatomic], [strong]
```

requestProblemInformation specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.22 requestResponseInformation

```
- (NSNumber*) requestResponseInformation  [read], [write], [nonatomic], [strong]
```

requestResponseInformation specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.23 runLoop

```
- (NSRunLoop*) runLoop  [read], [write], [nonatomic], [strong]
```

runLoop The runLoop where the streams are scheduled. If nil, defaults to [NSRunLoop currentRunLoop].

5.10.3.24 runLoopMode

```
- (NSString*) runLoopMode  [read], [write], [nonatomic], [strong]
```

runLoopMode The runLoopMode where the streams are scheduled. If nil, defaults to NSRunLoopCommonModes.

5.10.3.25 serverKeepAlive

```
- (NSNumber*) serverKeepAlive  [read], [nonatomic], [strong]
```

The serverKeepAlive is a time interval measured in seconds. This value may be set by the broker and overrides keepAliveInterval if present Zero means the broker does not perform any keep alive checks

5.10.3.26 sessionExpiryInterval

```
- (NSNumber*) sessionExpiryInterval  [read], [write], [nonatomic], [strong]
```

sessionExpiryInterval specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.27 sessionPresent

```
- (BOOL) sessionPresent  [read], [nonatomic], [assign]
```

Indicates if the broker found a persistent session when connecting with cleanSession:FALSE

5.10.3.28 status

```
- (MQTTSessionStatus) status  [read], [nonatomic], [assign]
```

Session status

5.10.3.29 topicAliasMaximum

```
- (NSNumber*) topicAliasMaximum  [read], [write], [nonatomic], [strong]
```

topicAliasMaximum specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.30 transport

```
- (id<MQTTTransport>) transport  [read], [write], [nonatomic], [strong]
```

the transport provider for MQTTClient

assign an in instance of a class implementing the MQTTTransport protocol e.g. MQTTCFSocketTransport before connecting.

5.10.3.31 userName

```
- (NSString*) userName  [read], [write], [nonatomic], [strong]
```

see userName an NSString object containing the user's name (or ID) for authentication. May be nil.

5.10.3.32 userProperty

```
- (NSDictionary<NSString *, NSString*>*) userProperty  [read], [write], [nonatomic], [strong]
```

topicAliasMaximum specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.33 voip

```
- (BOOL) voip  [read], [write], [nonatomic], [assign]
```

Require for VoIP background service defaults to NO

5.10.3.34 willDelayInterval

```
- (NSNumber*) willDelayInterval  [read], [write], [nonatomic], [strong]
```

willDelayInterval specifies the number of seconds after which a session should expire MQTT v5.0

5.10.3.35 willFlag

```
- (BOOL) willFlag  [read], [write], [nonatomic], [assign]
```

willFlag If the Will Flag is set to YES this indicates that a Will Message MUST be published by the Server when the Server detects that the Client is disconnected for any reason other than the Client flowing a DISCONNECT Packet.

5.10.3.36 willMsg

```
- (NSData*) willMsg  [read], [write], [nonatomic], [strong]
```

willMsg If the Will Flag is set to YES the Will Message must be specified, nil otherwise.

5.10.3.37 willQoS

```
- (MQTTQosLevel) willQoS  [read], [write], [nonatomic], [assign]
```

willQoS specifies the QoS level to be used when publishing the Will Message. If the Will Flag is set to NO, then the Will QoS MUST be set to 0. If the Will Flag is set to YES, the Will QoS MUST be a valid MQTTQosLevel.

5.10.3.38 willRetainFlag

```
- (BOOL) willRetainFlag  [read], [write], [nonatomic], [assign]
```

willRetainFlag indicates if the server should publish the Will Messages with retainFlag. If the Will Flag is set to NO, then the Will Retain Flag MUST be set to NO . If the Will Flag is set to YES: If Will Retain is set to NO, the Serve MUST publish the Will Message as a non-retained publication [MQTT-3.1.2-14]. If Will Retain is set to YES, the Server MUST publish the Will Message as a retained publication [MQTT-3.1.2-15].

5.10.3.39 willTopic

```
- (NSString*) willTopic  [read], [write], [nonatomic], [strong]
```

willTopic If the Will Flag is set to YES, the Will Topic is a string, nil otherwise.

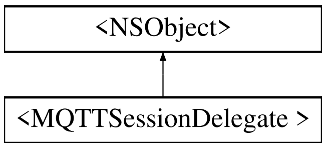
The documentation for this class was generated from the following file:

- MQTTSession.h

5.11 <MQTTSessionDelegate > Protocol Reference

```
#import <MQTTSession.h>
```

Inheritance diagram for <MQTTSessionDelegate >:



Instance Methods

- (void) - newMessage:data:onTopic:qos:retained:mid:
- (BOOL) - newMessageWithFeedback:data:onTopic:qos:retained:mid:
- (void) - session:newMessage:onTopic:
- (void) - handleEvent:event:error:
- (void) - session:handleEvent:
- (void) - connected:
- (void) - connected:sessionPresent:
- (void) - connectionRefused:error:
- (void) - connectionClosed:
- (void) - connectionError:error:
- (void) - protocolError:error:
- (void) - messageDelivered:msgID:
- (void) - messageDelivered:msgID:topic:data:qos:retainFlag:
- (void) - subAckReceived:msgID:grantedQoss:
- (void) - unsubAckReceived:msgID:
- (void) - sending:type:qos:retained:duped:mid:data:
- (void) - received:type:qos:retained:duped:mid:data:
- (BOOL) - ignoreReceived:type:qos:retained:duped:mid:data:
- (void) - buffered:queued:flowingIn:flowingOut:
- (void) - buffered:flowingIn:flowingOut:

5.11.1 Detailed Description

Session delegate gives your application control over the MQTTSession

Note

all callback methods are optional

5.11.2 Method Documentation

5.11.2.1 buffered:flowingIn:flowingOut:()

```
- (void MQTTSessionDelegate) buffered:
            (MQTTSession *) session
            flowingIn:(NSUInteger) flowingIn
            flowingOut:(NSUInteger) flowingOut   [optional]
```

gets called when the content of MQTTClients internal buffers change use for monitoring the completion of transmitted and received messages

Parameters

session	the MQTTSession reporting the change
flowingIn	the number of incoming messages not acknowledged by the MQTTClient yet
flowingOut	the number of outgoing messages not yet acknowledged by the MQTT broker

5.11.2.2 buffered:queued:flowingIn:flowingOut:()

```
- (void MQTTSessionDelegate) buffered:
            (MQTTSession *) session
            queued:(NSUInteger) queued
            flowingIn:(NSUInteger) flowingIn
            flowingOut:(NSUInteger) flowingOut   [optional]
```

gets called when the content of MQTTClients internal buffers change use for monitoring the completion of transmitted and received messages

Parameters

session	the MQTTSession reporting the change
queued	for backward compatibility only: MQTTClient does not queue messages anymore except during QoS protocol
flowingIn	the number of incoming messages not acknowledged by the MQTTClient yet
flowingOut	the number of outgoing messages not yet acknowledged by the MQTT broker

5.11.2.3 connected:()

```
- (void MQTTSessionDelegate) connected:
            (MQTTSession *) session   [optional]
```

gets called when a connection has been successfully established

Parameters

session	the MQTTSession reporting the connect

5.11.2.4 connected:sessionPresent:()

```
- (void MQTTSessionDelegate) connected:
            (MQTTSession *) session
            sessionPresent:(BOOL) sessionPresent   [optional]
```

gets called when a connection has been successfully established

Parameters

session	the MQTTSession reporting the connect
sessionPresent	represents the Session Present flag sent by the broker

5.11.2.5 connectionClosed:()

```
- (void MQTTSessionDelegate) connectionClosed:
            (MQTTSession *) session   [optional]
```

gets called when a connection has been closed

Parameters

session	the MQTTSession reporting the close

5.11.2.6 connectionError:error:()

```
- (void MQTTSessionDelegate) connectionError:
            (MQTTSession *) session
            error:(NSError *) error   [optional]
```

gets called when a connection error happened

Parameters

session	the MQTTSession reporting the connect error
error	an optional additional error object with additional information

5.11.2.7 connectionRefused:error:()

```
- (void MQTTSessionDelegate) connectionRefused:
            (MQTTSession *) session
            error:(NSError *) error   [optional]
```

gets called when a connection has been refused

Parameters

session	the MQTTSession reporting the refusal
error	an optional additional error object with additional information

5.11.2.8 handleEvent:event:error:()

```
- (void MQTTSessionDelegate) handleEvent:
              (MQTTSession *) session
              event:(MQTTSessionEvent) eventCode
              error:(NSError *) error   [optional]
```

gets called when a connection is established, closed or a problem occurred

Parameters

session	the MQTTSession reporting the event
eventCode	the code of the event
error	an optional additional error object with additional information

5.11.2.9 ignoreReceived:type:qos:retained:duped:mid:data:()

```
- (BOOL MQTTSessionDelegate) ignoreReceived:
              (MQTTSession *) session
              type:(MQTTCommandType) type
              qos:(MQTTQosLevel) qos
              retained:(BOOL) retained
              duped:(BOOL) duped
              mid:(UInt16) mid
              data:(NSData *) data   [optional]
```

gets called when a command is received from the MQTT broker use this for low level control of the MQTT connection

Parameters

session	the MQTTSession reporting the received command
type	the MQTT command type
qos	the Quality of Service of the command
retained	the retained status of the command
duped	the duplication status of the command
mid	the Message Identifier of the command
data	the payload data of the command if any, might be zero length

Returns

true if the sessionmanager should ignore the received message

5.11.2.10 messageDelivered:msgID:()

```
- (void MQTTSessionDelegate) messageDelivered:
             (MQTTSession *) session
             msgID:(UInt16) msgID   [optional]
```

gets called when a published message was actually delivered

Parameters

session	the MQTTSession reporting the delivery
msgID	the Message Identifier of the delivered message

Note

this method is called after a publish with qos 1 or 2 only

5.11.2.11 messageDelivered:msgID:topic:data:qos:retainFlag:()

```
- (void MQTTSessionDelegate) messageDelivered:
             (MQTTSession *) session
             msgID:(UInt16) msgID
             topic:(NSString *) topic
             data:(NSData *) data
             qos:(MQTTQosLevel) qos
             retainFlag:(BOOL) retainFlag   [optional]
```

gets called when a published message was actually delivered

Parameters

session	the MQTTSession reporting the delivery
msgID	the Message Identifier of the delivered message
topic	the topic of the delivered message
data	the data Identifier of the delivered message
qos	the QoS level of the delivered message
retainFlag	the retain Flag of the delivered message

Note

this method is called after a publish with qos 1 or 2 only

5.11.2.12 newMessage:data:onTopic:qos:retained:mid:()

```
- (void MQTTSessionDelegate) newMessage:
             (MQTTSession *) session
```

```
                data:(NSData *) data
                onTopic:(NSString *) topic
                qos:(MQTTQosLevel) qos
                retained:(BOOL) retained
                mid:(unsigned int) mid   [optional]
```

gets called when a new message was received

Parameters

session	the MQTTSession reporting the new message
data	the data received, might be zero length
topic	the topic the data was published to
qos	the qos of the message
retained	indicates if the data retransmitted from server storage
mid	the Message Identifier of the message if qos = 1 or 2, zero otherwise

5.11.2.13 newMessageWithFeedback:data:onTopic:qos:retained:mid:()

```
- (BOOL MQTTSessionDelegate) newMessageWithFeedback:
                (MQTTSession *) session
                data:(NSData *) data
                onTopic:(NSString *) topic
                qos:(MQTTQosLevel) qos
                retained:(BOOL) retained
                mid:(unsigned int) mid   [optional]
```

gets called when a new message was received

Parameters

session	the MQTTSession reporting the new message
data	the data received, might be zero length
topic	the topic the data was published to
qos	the qos of the message
retained	indicates if the data retransmitted from server storage
mid	the Message Identifier of the message if qos = 1 or 2, zero otherwise

Returns

true if the message was or will be processed, false if the message shall not be ack-ed

5.11.2.14 protocolError:error:()

```
- (void MQTTSessionDelegate) protocolError:
                (MQTTSession *) session
                error:(NSError *) error   [optional]
```

gets called when an MQTT protocol error happened

Parameters

session	the MQTTSession reporting the protocol error
error	an optional additional error object with additional information

5.11.2.15 received:type:qos:retained:duped:mid:data:()

```
- (void MQTTSessionDelegate) received:
            (MQTTSession *) session
            type:(MQTTCommandType) type
            qos:(MQTTQosLevel) qos
            retained:(BOOL) retained
            duped:(BOOL) duped
            mid:(UInt16) mid
            data:(NSData *) data   [optional]
```

gets called when a command is received from the MQTT broker use this for low level monitoring of the MQTT connection

Parameters

session	the MQTTSession reporting the received command
type	the MQTT command type
qos	the Quality of Service of the command
retained	the retained status of the command
duped	the duplication status of the command
mid	the Message Identifier of the command
data	the payload data of the command if any, might be zero length

5.11.2.16 sending:type:qos:retained:duped:mid:data:()

```
- (void MQTTSessionDelegate) sending:
            (MQTTSession *) session
            type:(MQTTCommandType) type
            qos:(MQTTQosLevel) qos
            retained:(BOOL) retained
            duped:(BOOL) duped
            mid:(UInt16) mid
            data:(NSData *) data   [optional]
```

gets called when a command is sent to the MQTT broker use this for low level monitoring of the MQTT connection

Parameters

session	the MQTTSession reporting the sent command
type	the MQTT command type

Parameters

qos	the Quality of Service of the command
retained	the retained status of the command
duped	the duplication status of the command
mid	the Message Identifier of the command
data	the payload data of the command if any, might be zero length

5.11.2.17 session:handleEvent:()

```
- (void MQTTSessionDelegate) session:
            (MQTTSession *) session
            handleEvent:(MQTTSessionEvent) eventCode   [optional]
```

for mqttio-OBJC backward compatibility

Parameters

session	the MQTTSession reporting the event
eventCode	the code of the event

5.11.2.18 session:newMessage:onTopic:()

```
- (void MQTTSessionDelegate) session:
            (MQTTSession *) session
            newMessage:(NSData *) data
            onTopic:(NSString *) topic   [optional]
```

for mqttio-OBJC backward compatibility

Parameters

session	see newMessage for description
data	see newMessage for description
topic	see newMessage for description

5.11.2.19 subAckReceived:msgID:grantedQoss:()

```
- (void MQTTSessionDelegate) subAckReceived:
            (MQTTSession *) session
            msgID:(UInt16) msgID
            grantedQoss:(NSArray< NSNumber * > *) qoss   [optional]
```

gets called when a subscription is acknowledged by the MQTT broker

Parameters

session	the MQTTSession reporting the acknowledge
msgID	the Message Identifier of the SUBSCRIBE message
qoss	an array containing the granted QoS(s) related to the SUBSCRIBE message (see subscribeTopic, subscribeTopics)

5.11.2.20 unsubAckReceived:msgID:()

```
- (void MQTTSessionDelegate) unsubAckReceived:
            (MQTTSession *) session
            msgID:(UInt16) msgID   [optional]
```

gets called when an unsubscribe is acknowledged by the MQTT broker

Parameters

session	the MQTTSession reporting the acknowledge
msgID	the Message Identifier of the UNSUBSCRIBE message

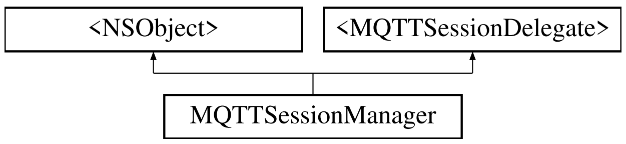
The documentation for this protocol was generated from the following file:

- MQTTSession.h

5.12 MQTTSessionManager Class Reference

```
#import <MQTTSessionManager.h>
```

Inheritance diagram for MQTTSessionManager:



Instance Methods

- (MQTTSessionManager *) - initWithPersistence:maxWindowSize:maxMessages:maxSize:maxConnectionRetryInterval:connectInForeground:
- (MQTTSessionManager *) - initWithPersistence:maxWindowSize:maxMessages:maxSize:connectInForeground:
- (MQTTSessionManager *) - initWithPersistence:maxWindowSize:maxMessages:maxSize:
- (void) - connectTo:port:tls:keepalive:clean:auth:user:pass:will:willTopic:willMsg:willQos:willRetainFlag:withClientId:securityPolicy:certificates:protocolLevel:

- (void) - connectTo:port:tls:keepalive:clean:auth:user:pass:will:willTopic:willMsg:willQos:willRetainFlag:with↩ClientId:securityPolicy:certificates:
- (void) - connectTo:port:tls:keepalive:clean:auth:user:pass:will:willTopic:willMsg:willQos:willRetainFlag:with↩ClientId:
- (void) - connectTo:port:tls:keepalive:clean:auth:user:pass:willTopic:will:willQos:willRetainFlag:withClientId:
- (void) - connectToLast
- (UInt16) - sendData:topic:qos:retain:
- (void) - disconnect

Properties

- MQTTSession ∗ session
- NSString ∗ host
- UInt32 port
- id< MQTTSessionManagerDelegate > delegate
- NSDictionary< NSString ∗, NSNumber ∗ > ∗ subscriptions
- NSDictionary< NSString ∗, NSNumber ∗ > ∗ effectiveSubscriptions
- MQTTSessionManagerState state
- NSError ∗ lastErrorCode

5.12.1 Detailed Description

SessionManager handles the MQTT session for your application

5.12.2 Method Documentation

5.12.2.1 connectTo:port:tls:keepalive:clean:auth:user:pass:will:willTopic:willMsg:willQos:willRetainFlag:withClientId:()

```
- (void) connectTo:
            (NSString *) host
            port:(NSInteger) port
            tls:(BOOL) tls
            keepalive:(NSInteger) keepalive
            clean:(BOOL) clean
            auth:(BOOL) auth
            user:(NSString *) user
            pass:(NSString *) pass
            will:(BOOL) will
            willTopic:(NSString *) willTopic
            willMsg:(NSData *) willMsg
            willQos:(MQTTQosLevel) willQos
            willRetainFlag:(BOOL) willRetainFlag
            withClientId:(NSString *) clientId
```

Convenience alternative to full paramter connectTo

Parameters

host	see connectTo description
port	see connectTo description
tls	see connectTo description
keepalive	see connectTo description
clean	see connectTo description
auth	see connectTo description
user	see connectTo description
pass	see connectTo description
will	see connectTo description
willTopic	see connectTo description
willMsg	see connectTo description
willQos	see connectTo description
willRetainFlag	see connectTo description
clientId	see connectTo description

5.12.2.2 connectTo:port:tls:keepalive:clean:auth:user:pass:will:willTopic:willMsg:willQos:willRetainFlag:withClientId:security↩ Policy:certificates:()

```
- (void) connectTo:
             (NSString *) host
             port:(NSInteger) port
             tls:(BOOL) tls
             keepalive:(NSInteger) keepalive
             clean:(BOOL) clean
             auth:(BOOL) auth
             user:(NSString *) user
             pass:(NSString *) pass
             will:(BOOL) will
             willTopic:(NSString *) willTopic
             willMsg:(NSData *) willMsg
             willQos:(MQTTQosLevel) willQos
             willRetainFlag:(BOOL) willRetainFlag
             withClientId:(NSString *) clientId
             securityPolicy:(MQTTSSLSecurityPolicy *) securityPolicy
             certificates:(NSArray *) certificates
```

Connects to the MQTT broker and stores the parameters for subsequent reconnects

Parameters

host	specifies the hostname or ip address to connect to. Defaults to "localhost".
port	specifies the port to connect to
tls	specifies whether to use SSL or not
keepalive	The Keep Alive is a time interval measured in seconds. The MQTTClient ensures that the interval between Control Packets being sent does not exceed the Keep Alive value. In the absence of sending any other Control Packets, the Client sends a PINGREQ Packet.
clean	specifies if the server should discard previous session information.
auth	specifies the user and pass parameters should be used for authenthication

Parameters

user	an NSString object containing the user's name (or ID) for authentication. May be nil.
pass	an NSString object containing the user's password. If userName is nil, password must be nil as well.
will	indicates whether a will shall be sent
willTopic	the Will Topic is a string, may be nil
willMsg	the Will Message, might be zero length or nil
willQos	specifies the QoS level to be used when publishing the Will Message.
willRetainFlag	indicates if the server should publish the Will Messages with retainFlag.
clientId	The Client Identifier identifies the Client to the Server. If nil, a random clientId is generated.
securityPolicy	A custom SSL security policy or nil.
certificates	An NSArray of the pinned certificates to use or nil.

5.12.2.3 connectTo:port:tls:keepalive:clean:auth:user:pass:will:willTopic:willMsg:willQos:willRetainFlag:withClientId:security↩ Policy:certificates:protocolLevel:()

```
- (void) connectTo:
            (NSString *) host
            port:(NSInteger) port
            tls:(BOOL) tls
            keepalive:(NSInteger) keepalive
            clean:(BOOL) clean
            auth:(BOOL) auth
            user:(NSString *) user
            pass:(NSString *) pass
            will:(BOOL) will
            willTopic:(NSString *) willTopic
            willMsg:(NSData *) willMsg
            willQos:(MQTTQosLevel) willQos
            willRetainFlag:(BOOL) willRetainFlag
            withClientId:(NSString *) clientId
            securityPolicy:(MQTTSSLSecurityPolicy *) securityPolicy
            certificates:(NSArray *) certificates
            protocolLevel:(MQTTProtocolVersion) protocolLevel
```

Connects to the MQTT broker and stores the parameters for subsequent reconnects

Parameters

host	specifies the hostname or ip address to connect to. Defaults to "localhost".
port	specifies the port to connect to
tls	specifies whether to use SSL or not
keepalive	The Keep Alive is a time interval measured in seconds. The MQTTClient ensures that the interval between Control Packets being sent does not exceed the Keep Alive value. In the absence of sending any other Control Packets, the Client sends a PINGREQ Packet.
clean	specifies if the server should discard previous session information.
auth	specifies the user and pass parameters should be used for authenthication
user	an NSString object containing the user's name (or ID) for authentication. May be nil.
pass	an NSString object containing the user's password. If userName is nil, password must be nil as well.

Parameters

will	indicates whether a will shall be sent
willTopic	the Will Topic is a string, may be nil
willMsg	the Will Message, might be zero length or nil
willQos	specifies the QoS level to be used when publishing the Will Message.
willRetainFlag	indicates if the server should publish the Will Messages with retainFlag.
clientId	The Client Identifier identifies the Client to the Server. If nil, a random clientId is generated.
securityPolicy	A custom SSL security policy or nil.
certificates	An NSArray of the pinned certificates to use or nil.
protocolLevel	Protocol version of the connection.

5.12.2.4 connectTo:port:tls:keepalive:clean:auth:user:pass:willTopic:will:willQos:willRetainFlag:withClientId:()

```
- (void) connectTo:
             (NSString *) host
             port:(NSInteger) port
             tls:(BOOL) tls
             keepalive:(NSInteger) keepalive
             clean:(BOOL) clean
             auth:(BOOL) auth
             user:(NSString *) user
             pass:(NSString *) pass
             willTopic:(NSString *) willTopic
             will:(NSData *) will
             willQos:(MQTTQosLevel) willQos
             willRetainFlag:(BOOL) willRetainFlag
             withClientId:(NSString *) clientId
```

Convenience alternative to full paramter connectTo

Parameters

host	see connectTo description
port	see connectTo description
tls	see connectTo description
keepalive	see connectTo description
clean	see connectTo description
auth	see connectTo description
user	see connectTo description
pass	see connectTo description
willTopic	the Will Topic is a string, must not be nil
will	the Will Message, might be zero length
willQos	see connectTo description
willRetainFlag	see connectTo description
clientId	see connectTo description

5.12.2.5 connectToLast()

```
- (void) connectToLast
```

Re-Connects to the MQTT broker using the parameters for given in the connectTo method

5.12.2.6 disconnect()

```
- (void) disconnect
```

Disconnects gracefully from the MQTT broker

5.12.2.7 initWithPersistence:maxWindowSize:maxMessages:maxSize:()

```
- (MQTTSessionManager *) initWithPersistence:
            (BOOL) persistent
            maxWindowSize:(NSUInteger) maxWindowSize
            maxMessages:(NSUInteger) maxMessages
            maxSize:(NSUInteger) maxSize
```

initWithPersistence sets the MQTTPersistence properties other than default

Parameters

persistent	YES or NO (default) to establish file or in memory persistence.
maxWindowSize	(a positive number, default is 16) to control the number of messages sent before waiting for acknowledgement in Qos 1 or 2. Additional messages are stored and transmitted later.
maxSize	(a positive number of bytes, default is 64 MB) to limit the size of the persistence file. Messages published after the limit is reached are dropped.
maxMessages	(a positive number, default is 1024) to limit the number of messages stored. Additional messages published are dropped.

Returns

the initialized MQTTSessionManager object

5.12.2.8 initWithPersistence:maxWindowSize:maxMessages:maxSize:connectInForeground:()

```
- (MQTTSessionManager *) initWithPersistence:
            (BOOL) persistent
            maxWindowSize:(NSUInteger) maxWindowSize
            maxMessages:(NSUInteger) maxMessages
            maxSize:(NSUInteger) maxSize
            connectInForeground:(BOOL) connectInForeground
```

initWithPersistence sets the MQTTPersistence properties other than default

Parameters

persistent	YES or NO (default) to establish file or in memory persistence.
maxWindowSize	(a positive number, default is 16) to control the number of messages sent before waiting for acknowledgement in Qos 1 or 2. Additional messages are stored and transmitted later.
maxSize	(a positive number of bytes, default is 64 MB) to limit the size of the persistence file. Messages published after the limit is reached are dropped.
maxMessages	(a positive number, default is 1024) to limit the number of messages stored. Additional messages published are dropped.
connectInForeground	Whether or not to connect the MQTTSession when the app enters the foreground, and disconnect when it becomes inactive. When NO, the caller is responsible for calling -connectTo: and -disconnect. Defaults to YES.

Returns

the initialized MQTTSessionManager object

5.12.2.9 initWithPersistence:maxWindowSize:maxMessages:maxSize:maxConnectionRetryInterval:connectInForeground:()

```
- (MQTTSessionManager *) initWithPersistence:
            (BOOL) persistent
            maxWindowSize:(NSUInteger) maxWindowSize
            maxMessages:(NSUInteger) maxMessages
            maxSize:(NSUInteger) maxSize
            maxConnectionRetryInterval:(NSTimeInterval) maxRetryInterval
            connectInForeground:(BOOL) connectInForeground
```

initWithPersistence sets the MQTTPersistence properties other than default

Parameters

persistent	YES or NO (default) to establish file or in memory persistence.
maxWindowSize	(a positive number, default is 16) to control the number of messages sent before waiting for acknowledgement in Qos 1 or 2. Additional messages are stored and transmitted later.
maxSize	(a positive number of bytes, default is 64 MB) to limit the size of the persistence file. Messages published after the limit is reached are dropped.
maxMessages	(a positive number, default is 1024) to limit the number of messages stored. Additional messages published are dropped.
maxRetryInterval	The duration at which the connection-retry timer should be capped. When MQTTSessionManager receives a ClosedByBroker or an Error event, it will attempt to reconnect to the broker. The time in between connection attempts is doubled each time, until it remains at maxRetryInterval. Defaults to 64 seconds.
connectInForeground	Whether or not to connect the MQTTSession when the app enters the foreground, and disconnect when it becomes inactive. When NO, the caller is responsible for calling -connectTo: and -disconnect. Defaults to YES.

Returns

the initialized MQTTSessionManager object

5.12.2.10 sendData:topic:qos:retain:()

```
- (UInt16) sendData:
             (NSData *) data
            topic:(NSString *) topic
            qos:(MQTTQosLevel) qos
            retain:(BOOL) retainFlag
```

publishes data on a given topic at a specified QoS level and retain flag

Parameters

data	the data to be sent. length may range from 0 to 268,435,455 - 4 - *lengthof-topic* bytes. Defaults to length 0.
topic	the Topic to identify the data
retainFlag	if YES, data is stored on the MQTT broker until overwritten by the next publish with retainFlag = YES
qos	specifies the Quality of Service for the publish qos can be 0, 1, or 2.

Returns

the Message Identifier of the PUBLISH message. Zero if qos 0. If qos 1 or 2, zero if message was dropped

Note

returns immediately.

5.12.3 Property Documentation

5.12.3.1 delegate

```
- (id<MQTTSessionManagerDelegate>) delegate  [read], [write], [nonatomic], [weak]
```

the delegate receiving incoming messages

5.12.3.2 effectiveSubscriptions

```
- (NSDictionary<NSString *, NSNumber *>*) effectiveSubscriptions  [read], [nonatomic], [strong]
```

effectiveSubscriptions s a dictionary of NSNumber instances indicating the granted MQTTQoSLevel, or 0x80 for subscription failure. The keys are topic filters. effectiveSubscriptions is observable and is updated everytime subscriptions change

```
    ...
    MQTTSessionManager *manager = [[MQTTSessionManager alloc] init];
    manager.delegate = self;

    [manager addObserver:self
        forKeyPath:@"effectiveSubscriptions"
        options:NSKeyValueObservingOptionInitial | NSKeyValueObservingOptionNew
        context:nil];
        manager.subscriptions = [@{@"#": @(0)} mutableCopy];
        [manager connectTo: ...
    ...
    [manager removeObserver:self forKeyPath:@"effectiveSubscriptions"];
    ...

- (void)observeValueForKeyPath:(NSString *)keyPath
    ofObject:(id)object
    change:(NSDictionary<NSString *,id> *)change
    context:(void *)context {
    if ([keyPath isEqualToString:@"effectiveSubscriptions"]) {
        MQTTSessionManager *manager = (MQTTSessionManager *)object;
        DDLogVerbose(@"effectiveSubscriptions changed: %@", manager.
      effectiveSubscriptions);
    }
}
```

5.12.3.3 host

```
- (NSString*) host  [read], [atomic], [assign]
```

host an NSString containing the hostName or IP address of the Server

5.12.3.4 lastErrorCode

```
- (NSError*) lastErrorCode  [read], [nonatomic], [assign]
```

SessionManager last error code when state equals MQTTSessionManagerStateError

5.12.3.5 port

```
- (UInt32) port  [read], [atomic], [assign]
```

port an unsigned 32 bit integer containing the IP port number of the Server

5.12.3.6 session

```
- (MQTTSession*) session  [read], [nonatomic], [strong]
```

Underlying MQTTSession currently in use.

5.12.3.7 state

```
- (MQTTSessionManagerState) state  [read], [nonatomic], [assign]
```

SessionManager status

5.12.3.8 subscriptions

```
- (NSDictionary<NSString *, NSNumber *>*) subscriptions  [read], [write], [nonatomic], [strong]
```

subscriptions is a dictionary of NSNumber instances indicating the MQTTQoSLevel. The keys are topic filters. The SessionManager subscribes to the given subscriptions after successfull (re-)connect according to the cleansession parameter and the state of the session as indicated by the broker. Setting a new subscriptions dictionary initiates SUBSCRIBE or UNSUBSCRIBE messages by SessionManager by comparing the old and new subscriptions.

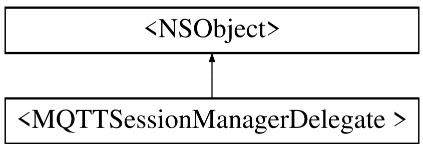
The documentation for this class was generated from the following file:

- MQTTSessionManager.h

5.13 <MQTTSessionManagerDelegate > Protocol Reference

```
#import <MQTTSessionManager.h>
```

Inheritance diagram for <MQTTSessionManagerDelegate >:



Instance Methods

- (void) - handleMessage:onTopic:retained:
- (void) - sessionManager:didReceiveMessage:onTopic:retained:
- (void) - messageDelivered:
- (void) - sessionManager:didDeliverMessage:
- (void) - sessionManager:didChangeState:

5.13.1 Detailed Description

delegate gives your application access to received messages

5.13.2 Method Documentation

5.13.2.1 handleMessage:onTopic:retained:()

```
- (void MQTTSessionManagerDelegate) handleMessage:
            (NSData *) data
            onTopic:(NSString *) topic
            retained:(BOOL) retained   [optional]
```

gets called when a new message was received

Parameters

data	the data received, might be zero length
topic	the topic the data was published to
retained	indicates if the data retransmitted from server storage

5.13.2.2 messageDelivered:()

```
- (void MQTTSessionManagerDelegate) messageDelivered:
                (UInt16) msgID   [optional]
```

gets called when a published message was actually delivered

Parameters

msgID	the Message Identifier of the delivered message

Note

this method is called after a publish with qos 1 or 2 only

5.13.2.3 sessionManager:didChangeState:()

```
- (void MQTTSessionManagerDelegate) sessionManager:
                (MQTTSessionManager *) sessionManager
                didChangeState:(MQTTSessionManagerState) newState   [optional]
```

gets called when the connection status changes

Parameters

sessionManager	the instance of MQTTSessionManager whose state changed
newState	the new connection state of the sessionManager. This will be identical to `sessionManager.state`.

5.13.2.4 sessionManager:didDeliverMessage:()

```
- (void MQTTSessionManagerDelegate) sessionManager:
                (MQTTSessionManager *) sessionManager
                didDeliverMessage:(UInt16) msgID   [optional]
```

gets called when a published message was actually delivered

Parameters

sessionManager	the instance of MQTTSessionManager whose state changed
msgID	the Message Identifier of the delivered message

Note

this method is called after a publish with qos 1 or 2 only

5.13.2.5 sessionManager:didReceiveMessage:onTopic:retained:()

```
- (void MQTTSessionManagerDelegate) sessionManager:
            (MQTTSessionManager *) sessionManager
            didReceiveMessage:(NSData *) data
            onTopic:(NSString *) topic
            retained:(BOOL) retained  [optional]
```

gets called when a new message was received

Parameters

sessionManager	the instance of MQTTSessionManager whose state changed
data	the data received, might be zero length
topic	the topic the data was published to
retained	indicates if the data retransmitted from server storage

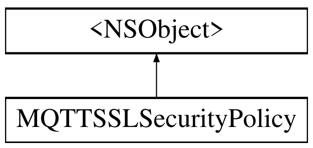
The documentation for this protocol was generated from the following file:

- MQTTSessionManager.h

5.14 MQTTSSLSecurityPolicy Class Reference

```
#import <MQTTSSLSecurityPolicy.h>
```

Inheritance diagram for MQTTSSLSecurityPolicy:



Instance Methods

Evaluating Server Trust

- (BOOL) - evaluateServerTrust:forDomain:

Class Methods

Getting Specific Security Policies

- (instancetype) + defaultPolicy

Initialization

- (instancetype) + policyWithPinningMode:

Properties

- MQTTSSLPinningMode SSLPinningMode
- BOOL validatesCertificateChain
- NSArray ∗ pinnedCertificates
- BOOL allowInvalidCertificates
- BOOL validatesDomainName

5.14.1 Detailed Description

`MQTTSSLSecurityPolicy` evaluates server trust against pinned X.509 certificates and public keys over secure connections.

If your app using security model which require pinning SSL certificates to helps prevent man-in-the-middle attacks and other vulnerabilities. you need to set securityPolicy to properly value(see MQTTSSLSecurityPolicy.h for more detail).

NOTE: about self-signed server certificates: if your server using Self-signed certificates to establish SSL/TLS connection, you need to set property: MQTTSSLSecurityPolicy.allowInvalidCertificates=YES.

If SSL is enabled, by default it only evaluate server's certificates using CA infrastructure, and for most case, this type of check is enough. However, if your app using security model which require pinning SSL certificates to helps prevent man-in-the-middle attacks and other vulnerabilities. you may need to set securityPolicy to properly value(see MQTTSSLSecurityPolicy.h for more detail).

NOTE: about self-signed server certificates: In CA infrastructure, you may establish a SSL/TLS connection with server which using self-signed certificates by install the certificates into OS keychain(either programmatically or manually). however, this method has some disadvantages:

1. every socket you app created will trust certificates you added.

2. if user choice to remove certificates from keychain, you app need to handling certificates re-adding.

If you only want to verify the cert for the socket you are creating and for no other sockets in your app, you need to use MQTTSSLSecurityPolicy. And if you use self-signed server certificates, your need to set property: MQTTSS←LSecurityPolicy.allowInvalidCertificates=YES

Adding pinned SSL certificates to your app helps prevent man-in-the-middle attacks and other vulnerabilities. Applications dealing with sensitive customer data or financial information are strongly encouraged to route all communication over an SSL/TLS connection with SSL pinning configured and enabled.

5.14.2 Method Documentation

5.14.2.1 defaultPolicy()

```
+ (instancetype) defaultPolicy
```

Returns the shared default security policy, which does not allow invalid certificates, validates domain name, and does not validate against pinned certificates or public keys.

Returns

The default security policy.

5.14.2.2 evaluateServerTrust:forDomain:()

```
- (BOOL) evaluateServerTrust:
            (SecTrustRef) serverTrust
            forDomain:(NSString *) domain
```

Whether or not the specified server trust should be accepted, based on the security policy.

This method should be used when responding to an authentication challenge from a server.

Parameters

serverTrust	The X.509 certificate trust of the server.
domain	The domain of serverTrust. If `nil`, the domain will not be validated.

Returns

Whether or not to trust the server.

5.14.2.3 policyWithPinningMode:()

```
+ (instancetype) policyWithPinningMode:
            (MQTTSSLPinningMode) pinningMode
```

Creates and returns a security policy with the specified pinning mode.

Parameters

pinningMode	The SSL pinning mode.

Returns

A new security policy.

5.14.3 Property Documentation

5.14.3.1 allowInvalidCertificates

```
- (BOOL) allowInvalidCertificates  [read], [write], [nonatomic], [assign]
```

Whether or not to trust servers with an invalid or expired SSL certificates. Defaults to `NO`. Note: If your server-certificates are self signed, your should set this property to 'YES'.

5.14.3.2 pinnedCertificates

```
- (NSArray*) pinnedCertificates  [read], [write], [nonatomic], [strong]
```

The certificates used to evaluate server trust according to the SSL pinning mode. By default, this property is set to any (`.cer`) certificates included in the app bundle. Note: Array item type: NSData - Bytes of X.509 certificate file in der format. Note that if you create an array with duplicate certificates, the duplicate certificates will be removed.

5.14.3.3 SSLPinningMode

```
- (MQTTSSLPinningMode) SSLPinningMode  [read], [nonatomic], [assign]
```

The criteria by which server trust should be evaluated against the pinned SSL certificates. Defaults to `MQTTSSL↩ PinningMode`.

5.14.3.4 validatesCertificateChain

```
- (BOOL) validatesCertificateChain  [read], [write], [nonatomic], [assign]
```

Whether to evaluate an entire SSL certificate chain, or just the leaf certificate. Defaults to `YES`.

5.14.3.5 validatesDomainName

```
- (BOOL) validatesDomainName  [read], [write], [nonatomic], [assign]
```

Whether or not to validate the domain name in the certificate's CN field. Defaults to `YES`.

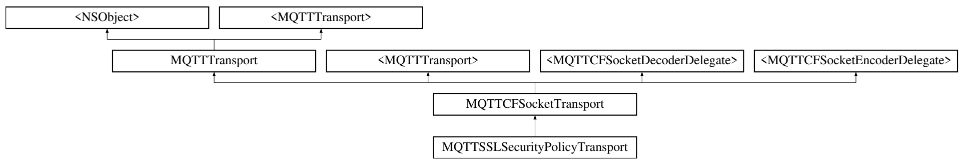
The documentation for this class was generated from the following file:

- MQTTSSLSecurityPolicy.h

5.15 MQTTSSLSecurityPolicyTransport Class Reference

```
#import <MQTTSSLSecurityPolicyTransport.h>
```

Inheritance diagram for MQTTSSLSecurityPolicyTransport:


<NSObject>
<MQTTTransport>
MQTTTransport
<MQTTTransport>
<MQTTCFSocketDecoderDelegate>
<MQTTCFSocketEncoderDelegate>
MQTTCFSocketTransport
MQTTSSLSecurityPolicyTransport

Properties

- MQTTSSLSecurityPolicy ∗ securityPolicy

Additional Inherited Members

5.15.1 Detailed Description

MQTTSSLSecurityPolicyTransport implements an extension of the MQTTCFSocketTransport by replacing the OS's certificate chain evaluation

5.15.2 Property Documentation

5.15.2.1 securityPolicy

```
- (MQTTSSLSecurityPolicy*) securityPolicy [read], [write], [nonatomic], [strong]
```

The security policy used to evaluate server trust for secure connections.

if your app using security model which require pinning SSL certificates to helps prevent man-in-the-middle attacks and other vulnerabilities. you need to set securityPolicy to properly value(see MQTTSSLSecurityPolicy.h for more detail).

NOTE: about self-signed server certificates: if your server using Self-signed certificates to establish SSL/TLS connection, you need to set property: MQTTSSLSecurityPolicy.allowInvalidCertificates=YES.

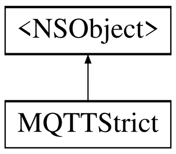
The documentation for this class was generated from the following file:

- MQTTSSLSecurityPolicyTransport.h

5.16 MQTTStrict Class Reference

```
#import <MQTTStrict.h>
```

Inheritance diagram for MQTTStrict:



Class Methods

- (BOOL) + strict
- (void) + setStrict:

5.16.1 Detailed Description

MQTTStrict controls the behaviour of MQTTClient with regards to parameter checking If strict is true, all parameters passed by the caller are checked before the corresponding message is send (CONNECT, PUBLISH, SUBSCRIBE, UNSUBSCRIBE) and an exception is thrown if any invalid values or inconsistencies are detected

If strict is false, parameters are used as passed by the caller. Messages will be sent "incorrectly" and parameter checking will be done on the broker end.

5.16.2 Method Documentation

5.16.2.1 setStrict:()

```
+ (void) setStrict:
                (BOOL) strict
```

setString sets the global strict flag

Parameters

strict	the new strict flag

5.16.2.2 strict()

```
+ (BOOL) strict
```

strict returns the current strict flag

Returns

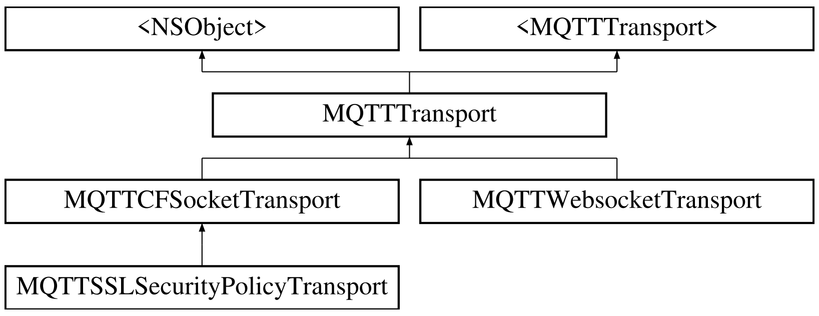
the strict flag

The documentation for this class was generated from the following file:

- MQTTStrict.h

5.17 MQTTTransport Class Reference

Inheritance diagram for MQTTTransport:



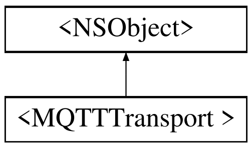
The documentation for this class was generated from the following file:

- MQTTTransport.h

5.18 <MQTTTransport > Protocol Reference

```
#import <MQTTTransport.h>
```

Inheritance diagram for <MQTTTransport >:



Instance Methods

- (void) - open
- (BOOL) - send:
- (void) - close

Properties

- NSRunLoop ∗_Nonnull runLoop
- NSString ∗_Nonnull runLoopMode
- NSString ∗_Nonnull host
- UInt32 port
- _Nullable id< MQTTTransportDelegate > delegate
- MQTTTransportState state

5.18.1 Detailed Description

MQTTTransport is a protocol abstracting the underlying transport level for MQTTClient

5.18.2 Method Documentation

5.18.2.1 close()

```
- (void MQTTTransport) close
```

close closes the transport

5.18.2.2 open()

```
- (void MQTTTransport) open
```

open opens the transport and prepares it for communication actual transports may require additional parameters to be set before opening

5.18.2.3 send:()

```
- (BOOL MQTTTransport) send:
                (nonnull NSData *) data
```

send transmits a data message

Parameters

data	data to be send, might be zero length

Returns

a boolean indicating if the data could be send or not

5.18.3 Property Documentation

5.18.3.1 delegate

```
- (_Nullable id<MQTTTransportDelegate> MQTTTransport) delegate [read], [write], [nonatomic],
[strong]
```

MQTTTransportDelegate needs to be set to a class implementing th MQTTTransportDelegate protocol to receive delegate messages.

5.18.3.2 host

```
- (NSString* _Nonnull MQTTTransport) host [read], [write], [nonatomic], [strong]
```

host an NSString containing the hostName or IP address of the host to connect to

5.18.3.3 port

```
- (UInt32 MQTTTransport) port [read], [write], [nonatomic], [assign]
```

port an unsigned 32 bit integer containing the IP port number to connect to

5.18.3.4 runLoop

```
- (NSRunLoop* _Nonnull MQTTTransport) runLoop [read], [write], [nonatomic], [strong]
```

runLoop The runLoop where the streams are scheduled. If nil, defaults to [NSRunLoop currentRunLoop].

5.18.3.5 runLoopMode

```
- (NSString* _Nonnull MQTTTransport) runLoopMode [read], [write], [nonatomic], [strong]
```

runLoopMode The runLoopMode where the streams are scheduled. If nil, defaults to NSRunLoopCommonModes.

5.18.3.6 state

```
- (MQTTTransportState MQTTTransport) state [read], [write], [nonatomic], [assign]
```

state contains the current MQTTTransportState of the transport

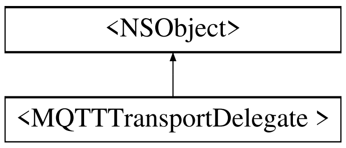
The documentation for this protocol was generated from the following file:

- MQTTTransport.h

5.19 <MQTTTransportDelegate > Protocol Reference

```
#import <MQTTTransport.h>
```

Inheritance diagram for <MQTTTransportDelegate >:



Instance Methods

- (void) - mqttTransport:didReceiveMessage:
- (void) - mqttTransportDidOpen:
- (void) - mqttTransport:didFailWithError:
- (void) - mqttTransportDidClose:

5.19.1 Detailed Description

MQTTTransportDelegate protocol Note: the implementation of the didReceiveMessage method is mandatory, the others are optional

5.19.2 Method Documentation

5.19.2.1 mqttTransport:didFailWithError:()

```
- (void MQTTTransportDelegate) mqttTransport:
            (_Nonnull id< MQTTTransport >) mqttTransport
            didFailWithError:(nullable NSError *) error   [optional]
```

didFailWithError gets called when an error was detected on the transport

Parameters

mqttTransport	the transport which detected the error
error	available error information, might be nil

5.19.2.2 mqttTransport:didReceiveMessage:()

```
- (void MQTTTransportDelegate) mqttTransport:
```

```
            (nonnull id< MQTTTransport >) mqttTransport
            didReceiveMessage:(nonnull NSData *) message
```

didReceiveMessage gets called when a message was received

Parameters

mqttTransport	the transport on which the message was received
message	the data received which may be zero length

5.19.2.3 mqttTransportDidClose:()

```
- (void MQTTTransportDelegate) mqttTransportDidClose:
            (_Nonnull id< MQTTTransport >) mqttTransport   [optional]
```

mqttTransportDidClose gets called when the transport closed

Parameters

mqttTransport	the transport which was closed

5.19.2.4 mqttTransportDidOpen:()

```
- (void MQTTTransportDelegate) mqttTransportDidOpen:
            (_Nonnull id< MQTTTransport >) mqttTransport   [optional]
```

mqttTransportDidOpen gets called when a transport is successfully opened

Parameters

mqttTransport	the transport which was successfully opened

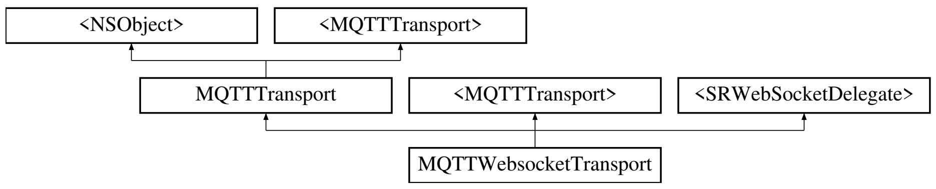
The documentation for this protocol was generated from the following file:

- MQTTTransport.h

5.20 MQTTWebsocketTransport Class Reference

```
#import <MQTTWebsocketTransport.h>
```

Inheritance diagram for MQTTWebsocketTransport:



Properties

- NSString ∗ host
- UInt32 port
- BOOL tls
- NSString ∗ path
- BOOL allowUntrustedCertificates
- NSArray ∗ pinnedCertificates

5.20.1 Detailed Description

MQTTCFSocketTransport implements an MQTTTransport on top of Websockets (SocketRocket)

5.20.2 Property Documentation

5.20.2.1 allowUntrustedCertificates

```
- (BOOL) allowUntrustedCertificates  [read], [write], [nonatomic], [assign]
```

allowUntrustedCertificates a boolean indicating whether self signed or expired certificates should be accepted defaults to NO

5.20.2.2 host

```
- (NSString∗) host  [read], [write], [nonatomic], [strong]
```

host an NSString containing the hostName or IP address of the host to connect to defaults to "localhost"

5.20.2.3 path

```
- (NSString∗) path  [read], [write], [nonatomic], [strong]
```

path an NSString indicating the path component of the websocket URL request defaults to "/html"

5.20.2.4 pinnedCertificates

```
- (NSArray*) pinnedCertificates  [read], [write], [nonatomic], [strong]
```

pinnedCertificates an NSArray containing certificates to validate server certificates against defaults to nil

5.20.2.5 port

```
- (UInt32) port  [read], [write], [nonatomic], [assign]
```

port an unsigned 32 bit integer containing the IP port number to connect to defaults to 80

5.20.2.6 tls

```
- (BOOL) tls  [read], [write], [nonatomic], [assign]
```

tls a boolean indicating whether the transport should be using security defaults to NO

The documentation for this class was generated from the following file:

- MQTTWebsocketTransport/MQTTWebsocketTransport.h

Index