ClientMqtt

August 18, 2026

ClientMqtt

Kind: Class

Source: `packages/microservices/client/client-mqtt.ts`

Part of: Microservices

`ClientMqtt` is a NestJS microservices client transport that communicates with MQTT brokers using publish/subscribe topics. It manages the MQTT connection lifecycle, derives request and response topic patterns, and handles broker events such as errors, reconnects, offline status, and disconnects.

Extends: `ClientProxy`

Methods

Method	Signature	Returns
`getRequestPattern`	`getRequestPattern(pattern: string)`	`string`
`getResponsePattern`	`getResponsePattern(pattern: string)`	`string`
`close`	`close()`	`void`
`connect`	`connect()`	`Promise<any>`
`mergeCloseEvent`	`mergeCloseEvent(instance: MqttClient, source$: Observable<T>)`	`Observable<T>`
`createClient`	`createClient()`	`MqttClient`
`registerErrorListener`	`registerErrorListener(client: MqttClient)`	`void`
`registerOfflineListener`	`registerOfflineListener(client: MqttClient)`	`void`
`registerReconnectListener`	`registerReconnectListener(client: MqttClient)`	`void`

Method	Signature	Returns
`registerDisconnectListener`	`registerDisconnectListener(client: MqttClient)`	`void`
`registerCloseListener`	`registerCloseListener(client: MqttClient)`	`void`
`registerConnectListener`	`registerConnectListener(client: MqttClient)`	`void`
`on`	`on(event: EventKey, callback: EventCallback)`	`void`
`unwrap`	`unwrap()`	`T`
`createResponseCallback`	`createResponseCallback()`	`(channel: string, buffer: Buffer) => any`
`publish`	`publish(partialPacket: ReadPacket, callback: (packet: WritePacket) => any)`	`() => void`
`dispatchEvent`	`dispatchEvent(packet: ReadPacket)`	`Promise<any>`
`unsubscribeFromChannel`	`unsubscribeFromChannel(channel: string)`	`void`
`initializeSerializer`	`initializeSerializer(options: MqttOptions['options'])`	`void`
`mergePacketOptions`	`mergePacketOptions(requestOptions: MqttRecordOptions)`	`MqttRecordOptions

Properties

Property	Type
`logger`	`any`
`subscriptionsCount`	`any`
`url`	`string`
`mqttClient`	`MqttClient
`connectionPromise`	`Promise
`isInitialConnection`	`any`
`isReconnecting`	`any`

Property	Type
`pendingEventListeners`	`Array<{ event: keyof MqttEvents; callback: MqttEvents[keyof MqttEvents]; }>`

Where it refuses work

- `ClientMqtt` stops the work with `Error` when `!this.mqttClient` — “Not initialized. Please call the "connect" method first.”.
- `ClientMqtt` stops the work with an early return when `this.mqttClient`.
- `ClientMqtt` stops the work with an early return when `err instanceof EmptyError`.
- `ClientMqtt` stops the work with an early return when `err.code === ECONNREFUSED || err.code === ENOTFOUND`.
- `ClientMqtt` stops the work with an early return when `!callback`.
- `ClientMqtt` stops the work with an early return when `isDisposed || err`.

When something fails

- `ClientMqtt` handles failure in 1 place: it turns it into a return value in all 1.

Diagram

```
graph LR
  App[Application Service] --> Client[ClientMqtt]
  Client --> Connect[connect()]
  Connect --> Broker[MQTT Broker]
  Client --> Request[getRequestPattern()]
  Client --> Response[getResponsePattern()]
  Request --> Broker
  Broker --> Response
  Client --> Events[Connection Event Listeners]
  Events --> Error[error]
  Events --> Offline[offline]
  Events --> Reconnect[reconnect]
  Events --> Disconnect[close]
```

AI Coding Instructions

- Use `connect()` before sending requests when the client is created manually; Nest-managed clients may connect through application lifecycle handling.

- Use `send()` for request/response communication and `emit()` for fire-and-forget MQTT events.
- Keep request and response pattern generation consistent; `getRequestPattern()` and `getResponsePattern()` are part of the transport correlation flow.
- Do not bypass lifecycle handling: call `close()` during shutdown to release MQTT connections and subscriptions.
- Preserve error, offline, reconnect, and disconnect listener registration when extending or modifying connection behavior.

Relationships

- IMPORTS → `Logger`
- IMPORTS → `loadPackage`
- IMPORTS → `isObject`