


Commit 010bca6

 **HJianBo** authored on Mar 5 Verified

Merge pull request #659 from emqx/fix/publish-protocol-error-handling











Fix malformed PUBLISH parsing and handle protocol errors safely

 master (#659) · 2.2.3 2.2.2

2 parents [573c231](#) + [12ac4f9](#) commit 010bca6 

 **8 files changed** +233 -35 lines changed

 Top 

- ▼  CocoaMQTTTests
 -  CocoaMQTTReaderProtocolErrorTests.swift
 -  FrameTests.swift
- ▼  Source
 -  CocoaMQTT.swift
 -  CocoaMQTT5.swift
 -  CocoaMQTTReader.swift
 -  FrameAuth.swift
 -  FrameDisconnect.swift
 -  FramePublish.swift

 **8 files changed** +233 -35 lines changed



▼ ...sts/CocoaMQTTReaderProtocolErrorTests.swift 

... @@ -0,0 +1,105 @@

1 + import Foundation

```
2 + import XCTest
3 + @testable import CocoaMQTT
4 +
5 + final class CocoaMQTTReaderProtocolErrorTests: XCTestCase {
6 +
7 +     private final class SocketSpy: CocoaMQTTSocketProtocol {
8 +         var enableSSL: Bool = false
9 +         private(set) var disconnectCount = 0
10 +
11 +         func setDelegate(_ theDelegate: CocoaMQTTSocketDelegate?,
12 +             delegateQueue: DispatchQueue?) {}
13 +         func connect(toHost host: String, onPort port: UInt16) throws {}
14 +         func connect(toHost host: String, onPort port: UInt16, withTimeout
15 +             timeout: TimeInterval) throws {}
16 +         func disconnect() { disconnectCount += 1 }
17 +         func readData(toLength length: UInt, withTimeout timeout: TimeInterval,
18 +             tag: Int) {}
19 +         func write(_ data: Data, withTimeout timeout: TimeInterval, tag: Int)
20 +             {}
21 +     }
22 +
23 +     private final class ReaderDelegateSpy: CocoaMQTTReaderDelegate {
24 +         private(set) var publishCount = 0
25 +         private(set) var disconnectCount = 0
26 +         private(set) var authCount = 0
27 +
28 +         func didReceive(_ reader: CocoaMQTTReader, connack: FrameConnAck) {}
29 +         func didReceive(_ reader: CocoaMQTTReader, publish: FramePublish) {
30 +             publishCount += 1 }
31 +         func didReceive(_ reader: CocoaMQTTReader, puback: FramePubAck) {}
32 +         func didReceive(_ reader: CocoaMQTTReader, pubrec: FramePubRec) {}
33 +         func didReceive(_ reader: CocoaMQTTReader, pubrel: FramePubRel) {}
34 +         func didReceive(_ reader: CocoaMQTTReader, pubcomp: FramePubComp) {}
35 +         func didReceive(_ reader: CocoaMQTTReader, suback: FrameSubAck) {}
36 +         func didReceive(_ reader: CocoaMQTTReader, unsuback: FrameUnsubAck) {}
37 +         func didReceive(_ reader: CocoaMQTTReader, pingresp: FramePingResp) {}
38 +         func didReceive(_ reader: CocoaMQTTReader, disconnect: FrameDisconnect)
39 +             { disconnectCount += 1 }
40 +         func didReceive(_ reader: CocoaMQTTReader, auth: FrameAuth) { authCount
41 +             += 1 }
```

```
35 +     }
36 +
37 +     func testMalformedPublishDisconnectsSocket() {
38 +         CocoaMQTTStorage()?.setMQTTVersion("3.1.1")
39 +
40 +         let socket = SocketSpy()
41 +         let delegate = ReaderDelegateSpy()
42 +         let reader = CocoaMQTTReader(socket: socket, delegate: delegate)
43 +
44 +         reader.headerReady(FrameType.publish.rawValue)
45 +         reader.lengthReady(0x06)
46 +         reader.payloadReady(Data([0x00, 0x00, 0x41, 0x41, 0x41, 0x41]))
47 +
48 +         XCTAssertEqual(socket.disconnectCount, 1)
49 +         XCTAssertEqual(delegate.publishCount, 0)
50 +     }
51 +
52 +     func testUnknownFrameTypeDisconnectsSocket() {
53 +         let socket = SocketSpy()
54 +         let delegate = ReaderDelegateSpy()
55 +         let reader = CocoaMQTTReader(socket: socket, delegate: delegate)
56 +
57 +         reader.headerReady(0x00)
58 +         reader.lengthReady(0x00)
59 +
60 +         XCTAssertEqual(socket.disconnectCount, 1)
61 +         XCTAssertEqual(delegate.publishCount, 0)
62 +     }
63 +
64 +     func testMQTT5DisconnectFrameDoesNotProtocolError() {
65 +         CocoaMQTTStorage()?.setMQTTVersion("5.0")
66 +
67 +         let socket = SocketSpy()
68 +         let delegate = ReaderDelegateSpy()
69 +         let reader = CocoaMQTTReader(socket: socket, delegate: delegate)
70 +
71 +         reader.headerReady(FrameType.disconnect.rawValue)
72 +         reader.lengthReady(0x00)
73 +
74 +         XCTAssertEqual(socket.disconnectCount, 0)
```

```

75 +         XCTAssertEqual(delegate.disconnectCount, 1)
76 +     }
77 +
78 +     func testMQTT5AuthFrameDoesNotProtocolError() {
79 +         CocoaMQTTStorage()?.setMQTTVersion("5.0")
80 +
81 +         let socket = SocketSpy()
82 +         let delegate = ReaderDelegateSpy()
83 +         let reader = CocoaMQTTReader(socket: socket, delegate: delegate)
84 +
85 +         reader.headerReady(FrameType.auth.rawValue)
86 +         reader.lengthReady(0x00)
87 +
88 +         XCTAssertEqual(socket.disconnectCount, 0)
89 +         XCTAssertEqual(delegate.authCount, 1)
90 +     }
91 +
92 +     func testMQTT311RejectsMQTT5OnlyDisconnectFrame() {
93 +         CocoaMQTTStorage()?.setMQTTVersion("3.1.1")
94 +
95 +         let socket = SocketSpy()
96 +         let delegate = ReaderDelegateSpy()
97 +         let reader = CocoaMQTTReader(socket: socket, delegate: delegate)
98 +
99 +         reader.headerReady(FrameType.disconnect.rawValue)
100 +         reader.lengthReady(0x00)
101 +
102 +         XCTAssertEqual(socket.disconnectCount, 1)
103 +         XCTAssertEqual(delegate.disconnectCount, 0)
104 +     }
105 + }

```

▼ CocoaMQTTTests/FrameTests.swift



```

↑... @@ -142,6 +142,28 @@ class FrameTests: XCTestCase {
142 142         XCTAssertEqual(f1?.payload().count, 0)
143 143     }
144 144
145 +     func testFramePublishRejectsZeroLengthTopic() {
146 +         CocoaMQTTStorage()?.setMQTTVersion("3.1.1")

```

```

147 +         let frame = FramePublish(packetFixedHeaderType:
      FrameType.publish.rawValue, bytes: [0x00, 0x00, 0x41, 0x41, 0x41, 0x41])
148 +         XCTAssertNil(frame)
149 +     }
150 +
151 +     func testFramePublishRejectsZeroLengthTopicInMQTT5WithoutAlias() {
152 +         CocoaMQTTStorage()?.setMQTTVersion("5.0")
153 +         defer { CocoaMQTTStorage()?.setMQTTVersion("3.1.1") }
154 +
155 +         let frame = FramePublish(packetFixedHeaderType:
      FrameType.publish.rawValue, bytes: [0x00, 0x00, 0x00])
156 +         XCTAssertNil(frame)
157 +     }
158 +
159 +     func testFramePublishAllowsZeroLengthTopicInMQTT5WithAlias() {
160 +         CocoaMQTTStorage()?.setMQTTVersion("5.0")
161 +         defer { CocoaMQTTStorage()?.setMQTTVersion("3.1.1") }
162 +
163 +         let frame = FramePublish(packetFixedHeaderType:
      FrameType.publish.rawValue, bytes: [0x00, 0x00, 0x03, 0x23, 0x00, 0x01])
164 +         XCTAssertEqual(frame?.topic, "")
165 +     }
166 +

```

```

145 167     func testFramePubAck() {
146 168
147 169         var puback = FramePubAck(msgid: 0x1010)

```



Source/CocoaMQTT.swift



@@ -794,4 +794,14 @@ extension CocoaMQTT: CocoaMQTTReaderDelegate {

```

794 794         delegate?.mqttDidReceivePong(self)
795 795         didReceivePong(self)
796 796     }
797 +
798 +     func didReceive(_ reader: CocoaMQTTReader, disconnect: FrameDisconnect) {
799 +         printWarning("Received DISCONNECT in MQTT 3.1.1 mode, closing socket")
800 +         internal_disconnect()
801 +     }
802 +
803 +     func didReceive(_ reader: CocoaMQTTReader, auth: FrameAuth) {

```

```

804 +     printWarning("Received AUTH in MQTT 3.1.1 mode, closing socket")
805 +     internal_disconnect()
806 + }
797 807 }

```

Source/CocoaMQTT5.swift

```

@@ -702,13 +702,15 @@ extension CocoaMQTT5: CocoaMQTTSocketDelegate {
702 702     extension CocoaMQTT5: CocoaMQTTReaderDelegate {
703 703
704 704         func didReceive(_ reader: CocoaMQTTReader, disconnect: FrameDisconnect) {
705 -             delegate?.mqtt5(self, didReceiveDisconnectReasonCode:
              disconnect.receiveReasonCode!)
706 -             didDisconnectReasonCode(self, disconnect.receiveReasonCode!)
705 +             let reasonCode = disconnect.receiveReasonCode ?? .normalDisconnection
706 +             delegate?.mqtt5(self, didReceiveDisconnectReasonCode: reasonCode)
707 +             didDisconnectReasonCode(self, reasonCode)
707 708         }
708 709
709 710         func didReceive(_ reader: CocoaMQTTReader, auth: FrameAuth) {
710 -             delegate?.mqtt5(self, didReceiveAuthReasonCode:
              auth.receiveReasonCode!)
711 -             didAuthReasonCode(self, auth.receiveReasonCode!)
711 +             let reasonCode = auth.receiveReasonCode ?? .success
712 +             delegate?.mqtt5(self, didReceiveAuthReasonCode: reasonCode)
713 +             didAuthReasonCode(self, reasonCode)
712 714         }
713 715
714 716         func didReceive(_ reader: CocoaMQTTReader, connack: FrameConnAck) {

```

Source/CocoaMQTTReader.swift

```

@@ -35,6 +35,10 @@ protocol CocoaMQTTReaderDelegate: AnyObject {
35 35         func didReceive(_ reader: CocoaMQTTReader, unsuback: FrameUnsubAck)
36 36
37 37         func didReceive(_ reader: CocoaMQTTReader, pingresp: FramePingResp)
38 +
39 +         func didReceive(_ reader: CocoaMQTTReader, disconnect: FrameDisconnect)
40 +
41 +         func didReceive(_ reader: CocoaMQTTReader, auth: FrameAuth)

```

```

38 42 }
39 43
40 44 class CocoaMQTTReader {
@@ -109,8 +113,7 @@ class CocoaMQTTReader {
109 113     private func frameReady() {
110 114
111 115         guard let frameType = FrameType(rawValue: UInt8(header & 0xF0)) else {
112 -             printError("Received unknown frame type, header: \(header), data:\
(data)")
113 -             readHeader()
116 +             protocolError("Received unknown frame type, header: \(header),
data:\(data)")
114 117             return
115 118         }
116 119
@@ -119,65 +122,95 @@ class CocoaMQTTReader {
119 122         switch frameType {
120 123             case .connack:
121 124                 guard let connack = FrameConnAck(packetFixedHeaderType: header,
bytes: data) else {
122 -                 printError("Reader parse \(frameType) failed, data: \(data)")
123 -                 break
125 +                 protocolError("Reader parse \(frameType) failed, data: \(
(data)")
126 +                 return
124 127             }
125 128             delegate?.didReceive(self, connack: connack)
126 129             case .publish:
127 130                 guard let publish = FramePublish(packetFixedHeaderType: header,
bytes: data) else {
128 -                 printError("Reader parse \(frameType) failed, data: \(data)")
129 -                 break
131 +                 protocolError("Reader parse \(frameType) failed, data: \(
(data)")
132 +                 return
130 133             }
131 134             delegate?.didReceive(self, publish: publish)
132 135             case .puback:

```

```

133 136         guard let puback = FramePubAck(packetFixedHeaderType: header,
      bytes: data) else {
134 -             printError("Reader parse \((frameType) failed, data: \((data)")
135 -             break
137 +             protocolError("Reader parse \((frameType) failed, data: \
      (data)")
138 +             return
136 139         }
137 140         delegate?.didReceive(self, puback: puback)
138 141         case .pubrec:
139 142         guard let pubrec = FramePubRec(packetFixedHeaderType: header,
      bytes: data) else {
140 -             printError("Reader parse \((frameType) failed, data: \((data)")
141 -             break
143 +             protocolError("Reader parse \((frameType) failed, data: \
      (data)")
144 +             return
142 145         }
143 146         delegate?.didReceive(self, pubrec: pubrec)
144 147         case .pubrel:
145 148         guard let pubrel = FramePubRel(packetFixedHeaderType: header,
      bytes: data) else {
146 -             printError("Reader parse \((frameType) failed, data: \((data)")
147 -             break
149 +             protocolError("Reader parse \((frameType) failed, data: \
      (data)")
150 +             return
148 151         }
149 152         delegate?.didReceive(self, pubrel: pubrel)
150 153         case .pubcomp:
151 154         guard let pubcomp = FramePubComp(packetFixedHeaderType: header,
      bytes: data) else {
152 -             printError("Reader parse \((frameType) failed, data: \((data)")
153 -             break
155 +             protocolError("Reader parse \((frameType) failed, data: \
      (data)")
156 +             return
154 157         }
155 158         delegate?.didReceive(self, pubcomp: pubcomp)
156 159         case .suback:

```

```

157 160         guard let frame = FrameSubAck(packetFixedHeaderType: header, bytes:
          data) else {
158 -             printError("Reader parse \((frameType) failed, data: \((data)")
159 -             break
161 +             protocolError("Reader parse \((frameType) failed, data: \
          (data)")
162 +             return
160 163         }
161 164         delegate?.didReceive(self, suback: frame)
162 165         case .unsuback:
163 166         guard let frame = FrameUnsubAck(packetFixedHeaderType: header,
          bytes: data) else {
164 -             printError("Reader parse \((frameType) failed, data: \((data)")
165 -             break
167 +             protocolError("Reader parse \((frameType) failed, data: \
          (data)")
168 +             return
166 169         }
167 170         delegate?.didReceive(self, unsuback: frame)
168 171         case .pingresp:
169 172         guard let frame = FramePingResp(packetFixedHeaderType: header,
          bytes: data) else {
170 -             printError("Reader parse \((frameType) failed, data: \((data)")
171 -             break
173 +             protocolError("Reader parse \((frameType) failed, data: \
          (data)")
174 +             return
172 175         }
173 176         delegate?.didReceive(self, pingresp: frame)
177 +         case .disconnect:
178 +             guard isMQTT5ProtocolVersion() else {
179 +                 protocolError("Reader received MQTT5-only frame \((frameType) in
          non-MQTT5 mode, data: \((data)")
180 +                 return
181 +             }
182 +             guard let frame = FrameDisconnect(packetFixedHeaderType: header,
          bytes: data) else {
183 +                 protocolError("Reader parse \((frameType) failed, data: \
          (data)")
184 +                 return

```

```

185 +         }
186 +         delegate?.didReceive(self, disconnect: frame)
187 +         case .auth:
188 +             guard isMQTT5ProtocolVersion() else {
189 +                 protocolError("Reader received MQTT5-only frame \((frameType) in
non-MQTT5 mode, data: \((data)")
190 +                 return
191 +             }
192 +             guard let frame = FrameAuth(packetFixedHeaderType: header, bytes:
data) else {
193 +                 protocolError("Reader parse \((frameType) failed, data: \
(data)")
194 +                 return
195 +             }
196 +             delegate?.didReceive(self, auth: frame)
174 197         default:
175 -             break
198 +             protocolError("Received unsupported frame type \((frameType), data:
\((data)")
199 +             return
176 200         }
177 201
178 202         readHeader()
179 203     }
180 204
205 +     private func protocolError(_ reason: String) {
206 +         printError(reason)
207 +         socket.disconnect()
208 +     }
209 +
210 +     private func isMQTT5ProtocolVersion() -> Bool {
211 +         return CocoaMQTTStorage()?.queryMQTTVersion() == "5.0"
212 +     }
213 +
181 214     private func reset() {
182 215         length = 0
183 216         multiply = 1

```



Source/FrameAuth.swift



```

↑...    @@ -70,8 +70,18 @@ extension FrameAuth {
70 70    extension FrameAuth: InitialWithBytes {
71 71
72 72        init?(packetFixedHeaderType: UInt8, bytes: [UInt8]) {
73 -
74 -            receiveReasonCode = CocoaMQTTAUTHReasonCode(rawValue: bytes[0])
73 +
74 +            var protocolVersion = ""
75 +            if let storage = CocoaMQTTStorage() {
76 +                protocolVersion = storage.queryMQTTVersion()
77 +            }
78 +            guard protocolVersion == "5.0" else {
79 +                return nil
80 +            }
81 +            if bytes.isEmpty {
82 +                receiveReasonCode = .success
83 +            } else {
84 +                receiveReasonCode = CocoaMQTTAUTHReasonCode(rawValue: bytes[0])
85         }
86
87     }

```

Source/FrameDisconnect.swift

```

↑...    @@ -109,9 +109,13 @@ extension FrameDisconnect: InitialWithBytes {
109 109        }
110 110
111 111        if protocolVersion == "5.0" {
112 -            if bytes.count > 0 {
112 +            if bytes.isEmpty {
113 +                receiveReasonCode = .normalDisconnection
114 +            } else {
113 115                receiveReasonCode = CocoaMQTTDISCONNECTReasonCode(rawValue:
114 116                bytes[0])
117 +            } else {
118 +                return nil
115 119        }
116 120    }
117 121
↓...

```

```

Source/FramePublish.swift
@@ -156,14 +156,14 @@ extension FramePublish: InitialWithBytes {
    156 156         return nil
    157 157     }
    158 158
    159 -         let len = UInt16(bytes[0]) << 8 + UInt16(bytes[1])
    159 +         let topicLength = Int(UInt16(bytes[0]) << 8 | UInt16(bytes[1]))
    160 +         let topicStart = 2
    161 +         let topicEnd = topicStart + topicLength
    160 162
    161 -         // 2 is packetFixedHeaderType length
    162 -         var pos = 2 + Int(len)
    163 -
    164 -         if bytes.count < pos {
    163 +         if bytes.count < topicEnd {
    165 164         return nil
    166 165     }
    166 +         var pos = topicEnd
    167 167
    168 168         // msgid
    169 169         if (packetFixedHeaderType & 0x06) >> 1 == CocoaMQTTQoS.qos0.rawValue {
@@ -189,6 +189,14 @@ extension FramePublish: InitialWithBytes {
    189 189         if data.propertyLength != 0 {
    190 190             pos += data.propertyLength!
    191 191         }
    192 +         if pos > bytes.count {
    193 +             return nil
    194 +         }
    195 +
    196 +         // MQTT 5.0: Topic Name may be empty only when Topic Alias is
    197 +         // present.
    197 +         if data.topic.isEmpty && data.topicAlias == nil {
    198 +             return nil
    199 +         }
    192 200
    193 201         // MQTT 5.0
    194 202         self.mqtt5Topic = data.topic
@@ -201,9 +209,13 @@ extension FramePublish: InitialWithBytes {
    201 209

```

```
202 210         } else {
203 211         // MQTT 3.1.1
204 -         if let data = NSString(bytes: [UInt8](bytes[2...(pos-1)]), length:
      Int(len), encoding: String.Encoding.utf8.rawValue) {
205 -         topic = data as String
212 +         guard topicLength > 0 else {
213 +         return nil
214 +         }
215 +         guard let recTopic = String(bytes: bytes[topicStart..<topicEnd],
      encoding: .utf8) else {
216 +         return nil
206 217         }
218 +         topic = recTopic
207 219     }
208 220
209 221     // payload
⋮
↓
```

Comments 0



Please [sign in](#) to comment.