MQTT-TLS profile of ACE
draft-sengul-ace-mqtt-tls-profile-03

Abstract

   This document specifies a profile for the ACE (Authentication and
   Authorization for Constrained Environments) to enable authorization
   in an MQTT-based publish-subscribe messaging system. Proof-of-
   possession keys, bound to OAuth2.0 access tokens, are used to
   authenticate and authorize publisher and subscriber clients. The
   protocol relies on TLS for confidentiality and server authentication.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on April 8, 2019.

Table of Contents

1. Introduction

This document specifies a profile for the ACE framework
[I-D.ietf-ace-oauth-authz]. In this profile, clients and a resource
server use MQTT to communicate. The protocol relies on TLS for
communication security between entities. The basic protocol
interactions follow MQTT v3.1 - the OASIS Standard
[MQTT-OASIS-Standard]. In addition, this document describes

improvements to the basic protocol with the new MQTT v5 - the OASIS
Committee Specification [MQTT-OASIS-Standard-v5] (e.g., improved
authentication exchange and error reporting). Both versions are
expected to be supported in practice, and therefore, covered in this
document.

MQTT is a publish-subscribe protocol and supports two types of client
operation: publish and subscribe. Once connected, a client can
publish to multiple topics, and subscribe to multiple topics;
however, for this document, these actions are described separately.
The MQTT broker is responsible for distributing messages published by
the publishers to the appropriate subscribers. Each publish message
contains a topic, which is used by the broker to filter the
subscribers for the message. Subscribers must subscribe to the
topics to receive the corresponding messages.

In this document, message topics are treated as resources. Clients
use an access token, bound to a key (the proof-of-possession key) to
authorize with the MQTT broker their connection and publish/subscribe
permissions to topics. In the context of this ACE profile, the MQTT
broker acts as the resource server. To provide communication
confidentiality and resource server authentication, TLS is used.

Clients use client authorization servers [I-D.ietf-ace-actors] to
obtain tokens from the authorization server. The communication
protocol between the client authorization server and the
authorization server is assumed to be HTTPS. Also, if the broker
supports token introspection, it is assumed to use HTTPS to
communicate with the authorization server. These interfaces MAY be
implemented using other protocols, e.g., CoAP or MQTT. This document
makes the same assumptions as the Section 4 of the ACE framework
[I-D.ietf-ace-oauth-authz] regarding client and RS registration with
the AS and establishing of keying material.

This document describes the authorization of the following exchanges
between publisher and subscriber clients, and the broker.

o Connection establishment between the clients and the broker

o Publish messages from the publishers to the broker, and from the
   broker to the subscribers

o Subscribe messages from the subscribers to the broker

In Section 2, these exchanges are described based on the MQTT v3.1 -
the OASIS Standard [MQTT-OASIS-Standard]. These exchanges are also
supported by the new MQTT v5 - the OASIS Committee Specification

[MQTT-OASIS-Standard-v5].   Section 3 describes how they may be
improved by the new MQTT v5.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in BCP
14 [RFC2119] [RFC8174], when, and only when, they appear in all
capitals, as shown here.

1.2. ACE-Related Terminology

The terminology for entities in the architecture is defined in OAuth
2.0 RFC 6749 [RFC6749] and ACE actors [I-D.ietf-ace-actors], such as
"Client" (C), "Resource Server" (RS) and "Authorization Server" (AS).

The term "endpoint" is used following its OAuth definition, to denote
resources such as /token and /introspect at the AS.

The term "Resource" is used to refer to an MQTT "topic name," which
is defined in Section 1.3. Hence, the "Resource Owner" is any entity
that can authoritatively speak for the "topic".

Certain security-related terms such as "authentication",
"authorization", "confidentiality", "(data) integrity", "message
authentication code", and "verify" are taken from RFC 4949 [RFC4949].

1.3. MQTT-Related Terminology

The document describes message exchanges as MQTT protocol
interactions. For additional information, please refer to the MQTT
v3.1 - the OASIS Standard [MQTT-OASIS-Standard] or the MQTT v5 - the
OASIS Committee Specification [MQTT-OASIS-Standard-v5].

Topic name
        The label attached to an application message, which is
        matched to a subscription.

Topic filter
        An expression that indicates interest in one or more topic
        names. Topic filters may include wildcards.

Subscription
        A subscription comprises a Topic filter and a maximum quality
        of service (QoS).

Application Message

The data carried by the MQTT protocol. The data has an
associated QoS level and a Topic name.

MQTT sends various control messages across a network connection. The
following is not an exhaustive list and the control packets that are
not relevant for authorization are not explained. These include, for
instance, the PUBREL and PUBCOMP packets used in the 4-step handshake
required for the QoS level 2.

CONNECT
Client request to connect to the broker. After a network
connection is established, this is the first packet sent by a
client.

CONNACK
The broker connection acknowledgment. The first packet sent
from the broker to a client is a CONNACK packet. CONNACK
packets contain return codes indicating either a success or
an error state to a client.

PUBLISH
Publish packet that can be sent from a client to the broker,
or from the broker to a client.

PUBACK
Response to PUBLISH packet with QoS level 1. PUBACK can be
sent from the broker to a client or a client to the broker.

PUBREC
Response to PUBLISH packet with QoS level 2. PUBREC can be
sent from the broker to a client or a client to the broker.

SUBSCRIBE
The client subscribe request.

SUBACK
Subscribe acknowledgment.

PINGREQ A ping request sent from a client to the broker. It signals
to the broker that the client is alive, and is used to
confirm that the broker is still alive.

[2](). Basic Protocol Interactions

This section describes the following exchanges between publisher and
subscriber clients, the broker, and the authorization server
according to the MQTT v3.1 - the OASIS Standard
[MQTT-OASIS-Standard]. These exchanges are compatible also with the

new MQTT v5 - the OASIS Committee Specification
[MQTT-OASIS-Standard-v5]. In addition, Section 3 describes how these
exchanges may be improved with the MQTT v5.

o Authorizing connection establishment between the clients and the
   broker

o Authorizing publish messages from the publishers to the broker,
   and from the broker to the subscribers

o Authorizing subscribe messages from the subscribers to the broker

Message topics are treated as resources. The publisher and
subscriber clients are assumed to have identified the topics of
interest out-of-band (topic discovery is not a feature of the MQTT
protocol).

A connection request carries a token specifying the permissions that
the client has (e.g., publish permission to a given topic). A
resource owner can pre-configure policies at the AS that give clients
publish or subscribe permissions to different topics.

2.1. Authorizing Connection Establishment

This section specifies how publishers and subscribers establish an
authorized connection to an MQTT broker. The token request and
response use the /token endpoint of the authorization server, as
specified in Section 6 of the ACE framework
[I-D.ietf-ace-oauth-authz].

Figure 1 shows the basic protocol flow during connection
establishment. The step (C), client onboarding, is out of the scope
of this document. Steps (E) and (F) are optional.

```
                            +----------------+
      +---(A) Token request----| Client         |
      |                        | Authorization  |
      |    +-(B) Access token-->| Server         |
      |    |                   |_____|
      |    |                          |
      |    |                (C) Client On-boarding
      |    |                          |
      |    |                  +---------v-----+
   +--v------------+          | Publisher or  |
   |               |          | Subscriber    |
   | Authorization |          |_____|
   | Server        |            |       ^
   |_____|            |       |
      |    ^          (D)Connection  (G)Connection
      |    |            request +     response
      |    |            access token  |
      |    |                  |       |
      |    |                +---v-------------+
      |    |                |    Broker       |
      |    +(E)Introspection-| Resource Server |
      |   request (optional) |                 |
      +-(F)Introspection---->|_____|
        response (optional)
```
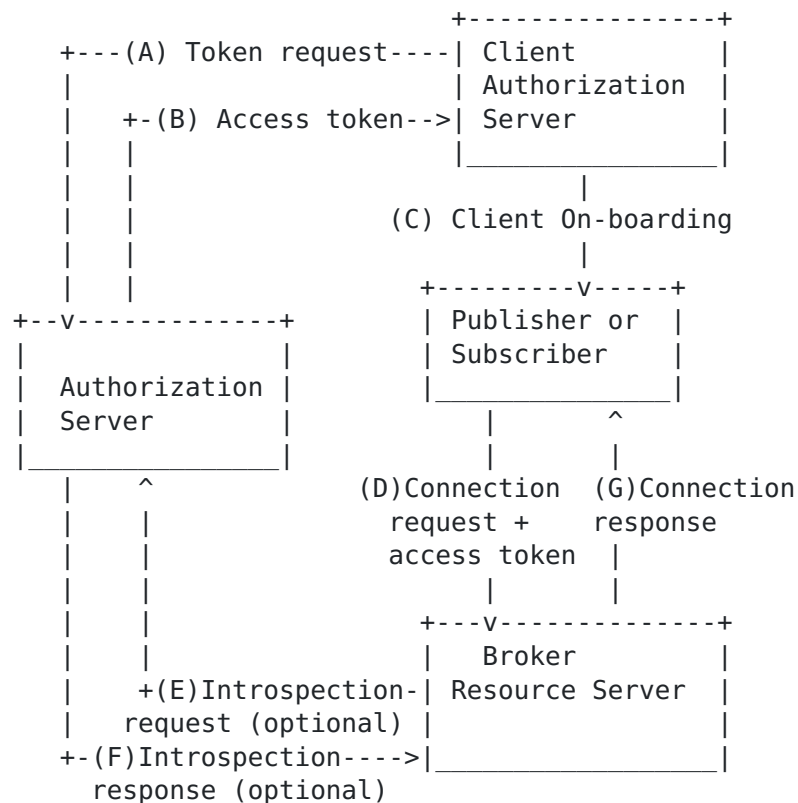
Figure 1: Connection establishment

2.1.1. Client Authorization Server (CAS) and Authorization Server (AS) Interaction

The first step in the protocol flow (Figure 1 (A)) is the token acquisition by the client authorization server (CAS) from the AS. If a client has enough resources and can support HTTPS, or optionally the AS supports MQTTS, these steps can instead be carried out by a client directly.

When requesting an access token from the AS, the CAS MAY include parameters in its request as defined in Section 6.1 of the ACE framework [I-D.ietf-ace-oauth-authz]. The content type is set to "application/json". The profile name is 'mqtt_tls'.

If the AS successfully verifies the access token request and authorizes the client for the indicated audience (e.g., RS) and scopes (e.g., publish/subscribe permissions over topics), the AS issues an access token (Figure 1 (B)). The response includes the parameters described in Section 6.2 of the ACE framework [I-D.ietf-ace-oauth-authz]. The included token is assumed to be Proof-of-Possession (PoP) token by default. Hence, a 'cnf' parameter

with a symmetric or asymmetric PoP key is returned. The token may be
a reference, or a CBOR or JWT web token. Note that the 'cnf'
parameter in the web tokens are to be consumed by the resource server
and not the client. For more information on Proof of Possession
semantics in JWTs see RFC 7800 [RFC7800] and for CWTs, see Proof-of-
Possession Key Semantics for CBOR Web Tokens (CWTs)
[I-D.ietf-ace-cwt-proof-of-possession].

In the case of an error, the AS returns error responses for HTTP-
based interactions as ASCII codes in JSON content, as defined in
Section 5.2 of RFC 6749 [RFC6749].

2.1.2. Client connection request to the broker

Once the client acquires the token, it can use it to request an MQTT
connection to the broker over a TLS session with server
authentication (Figure 1 (D)). This section describes the client
transporting the token to the broker (RS) via the CONNECT control
message after the TLS handshake. This is similar to an earlier
proposal by Fremantle et al. [fremantle14]. An improvement to this
is presented in Section 3 for the MQTT v5 - the OASIS Committee
Specification [MQTT-OASIS-Standard-v5]. Alternatively, the token may
be used for the TLS session establishment as described in the DTLS
profile for ACE [I-D.gerdes-ace-dtls-authorize]. In this case, both
the TLS PSK and RPK handshakes MAY be supported. This may
additionally require that the client transports the token to the
broker before the connection establishment. To this end, the broker
MAY support /authz-info endpoint via the "authz-info" topic. Then,
to transport the token, clients publish to "authz-info" topic
unauthorized. The topic "authz-info" MUST be publish-only for
clients (i.e., the clients are not allowed to subscribe to it). This
option is described in more detail in Appendix B.

When the client wishes to connect to the broker, it uses the CONNECT
message of MQTT. Figure 2 shows the structure of the MQTT CONNECT
control message.

```
       0            8            16           24           32
       +---------------------------------------------------------+
       |CPT=1 | Rsvd.|Remaining len.| Protocol  name len. = 4 |
       +---------------------------------------------------------+
       |                    'M' 'Q' 'T' 'T'                    |
       +---------------------------------------------------------+
       | Proto.level=4|Connect flags|        Keep alive      |
       +---------------------------------------------------------+
       | Payload                                               |
       |     Username as access token (UTF-8)                  |
       |     Password length (2 Bytes)                         |
       |     Password data as signature/MAC (binary)           |
       |                      ...                              |
       +---------------------------------------------------------+
```
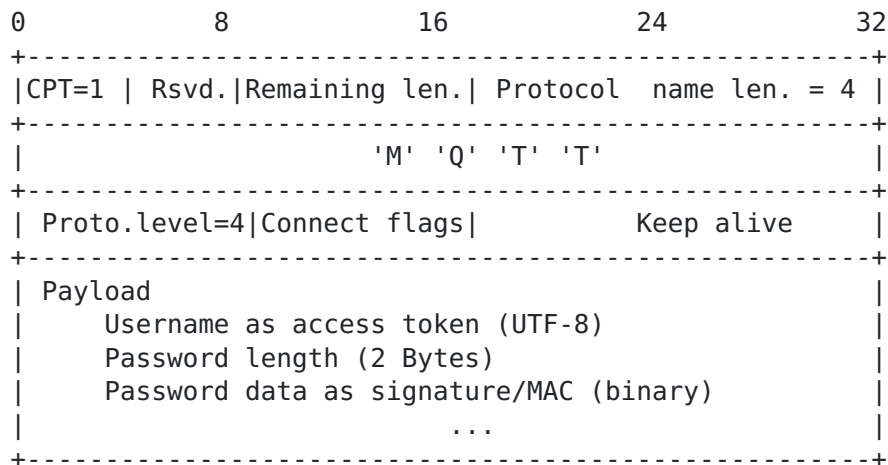
Figure 2: MQTT CONNECT control message. (CPT=Control Packet Type,
          Rsvd=Reserved, len.=length, Proto.=Protocol)

To communicate the necessary connection parameters, the Client uses
the appropriate flags of the CONNECT message. Figure 3 shows how the
MQTT connect flags MUST be set to initiate a connection with the
broker.

```
+-----------------------------------------------------------------+
|User name|Pass.|Will retain|Will QoS|Will Flag|Clean| Rsvd.|
| flag    |flag |           |        |         |     |      |
+-----------------------------------------------------------------+
| 1       | 1   |    X      |  X X   |    X    |  1  |  0   |
+-----------------------------------------------------------------+
```
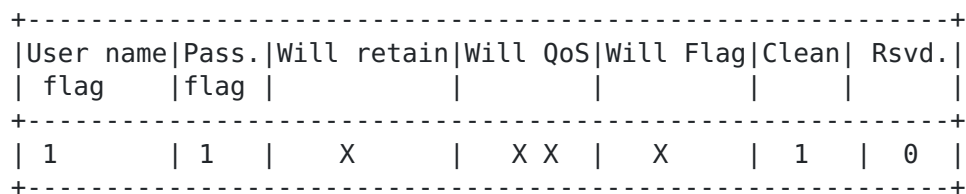
Figure 3: MQTT CONNECT flags. (Rsvd=Reserved)

To ensure that the client and the broker discard any previous session
and start a new session, the Clean Session Flag MUST be set to 1.

The Will flag indicates that a Will message needs to be sent when a
client disconnection occurs. The situations in which the Will
message is published include disconnections due to I/O or network
failures, and the server closing the networking connection due to a
protocol error. The client may set the Will flag as desired (marked
as 'X' in Figure 3). If the Will flag is set to 1 and the broker
accepts the connection request, the broker must store the Will
message, and publish it when the network connection is closed
according to Will QoS and Will retain parameters, and MQTT Will
management rules. Section 2.5 explains how the broker deals with the
retained messages in further detail.

Finally, Username and Password flags MUST be set to 1 to ensure that
the Payload of the CONNECT message includes both Username and
Password fields.

The CONNECT message defaults to ACE for authentication and
authorization. For the basic operation described in this section,
the Username field MUST be set to the access token. The Password
field MUST be set to the keyed message digest (MAC) or signature
associated with the access token for proof-of-possession. The client
MAY apply the PoP key either to the token or the entire request by
computing a keyed message digest (for symmetric key) or a digital
signature (for asymmetric key).

Section 3.1.3 of MQTT v3.1 - the OASIS Standard [MQTT-OASIS-Standard]
defines the MQTT Username as a UTF-8 encoded string, which is
prefixed by a 2-byte length field followed by UTF-8 encoded character
data up to 65535 bytes. Therefore an access token that is not valid
UTF-8 MUST be Base64 [RFC4648] encoded. (The MQTT Password allows
binary data (up to 65535 bytes) so does not require encoding.)

2.1.3. Token validation

RS MUST verify the validity of the token. This validation MAY be
done locally (e.g., in the case of a self-contained token) or the RS
MAY send an introspection request to the AS. If introspection is
used, this section follows similar steps to those described in
Sections 7.2 and 7.3 of the ACE framework [I-D.ietf-ace-oauth-authz].
The communication between AS and RS MAY be HTTPS, but it, in every
case, MUST be confidential, mutually authenticated and integrity
protected.

The broker MUST check if the token is active either using 'exp' claim
of the token or 'active' parameter of the introspection response.

The access token is constructed by the AS such that RS can associate
the access token with the client key. This document assumes that the
Access Token is a PoP token as described in
[I-D.ietf-ace-oauth-authz]. Therefore, the necessary information is
contained in the 'cnf' claim of the access token and may use either
public or shared key approaches. The client uses the signature or
the MAC in the password field to prove the possession of the key.
Depending on the chosen implementation, the resource server validates
the signature or the MAC over the token or the contents of the
packet, authenticating the client.

The broker uses the scope field in the token (or in the introspection
result) to determine the publish and subscribe permissions for the

client. If the Will flag is set, then the broker MUST check that the token allows the publication of the Will message too.

If the token is not self-contained and the broker uses token introspection, it MAY cache the validation result to decide whether to accept subsequent PUBLISH and SUBSCRIBE messages as these messages, which are sent after a connection is set-up, do not contain access tokens. If the introspection result is not cached, then the RS needs to introspect the saved token for each request.

Scope strings SHOULD be encoded as a permission, followed by an underscore, followed by a topic filter. Two permissions apply to topics: 'publish' and 'subscribe'. An example scope field may contain multiple such strings, space delimited, e.g., 'publish_topic1 subscribe_topic2/#'. Hence, this access token would give 'publish' permission to the 'topic1', 'subscribe' permission to all the subtopics of 'topic2'. If there is a single RS, then scope strings MAY be simply the keywords 'publish' or 'subscribe,' and the 'aud' field in the access token MAY be used to define the topic filter.

Also, if present in the access token, RS must check that the 'iss' corresponds to AS, the 'aud' field (if not used to define topics) corresponds to RS. It also has to check whether 'nbf' and 'iat' claims are present and valid.

2.1.4. The broker's response to client connection request

Based on the validation result (obtained either via local inspection or using the /introspection interface of the AS), the broker MUST send a CONNACK message to the client.

The broker responses may follow either the MQTT v3.1 - the OASIS Standard [MQTT-OASIS-Standard] or the MQTT v5 - the OASIS Committee Specification [MQTT-OASIS-Standard-v5], depending on which version(s) the broker supports.

In MQTT v3.1 - the OASIS Standard [MQTT-OASIS-Standard], it is not possible to support AS discovery via sending a tokenless CONNECT message to the broker. This is because a CONNACK packet does not include a means to provide additional information to the client. Therefore, AS discovery needs to take place out-of-band. This is remedied in the MQTT v5 - the OASIS Committee Specification [MQTT-OASIS-Standard-v5] and a solution is described in Section 3.

If the RS accepts the connection, it MUST store the token.

2.2. Authorizing PUBLISH messages

2.2.1. PUBLISH messages from the publisher client to the broker

On receiving the PUBLISH message, the broker MUST use the type of
message (i.e., PUBLISH) and the topic name in the message header to
compare against the cached token or its introspection result
(depending on the implementation, different fields of the token or
the introspection result may be checked, see Section 2.1.3).

If the client is allowed to publish to the topic, the RS must publish
the message to all valid subscribers of the topic. The broker may
also return an acknowledgment message if the QoS level is greater
than or equal to 1.

In case of a failure, it is not possible to return an error in MQTT
v3.1 - the OASIS Standard [MQTT-OASIS-Standard]. Acknowledgement
messages only indicate success. In the case of an authorization
error, the broker SHOULD disconnect the client. Otherwise, it MUST
ignore the PUBLISH message. Also, DISCONNECT messages are only sent
from a client to the broker. So, server disconnection needs to take
place below the application layer. However, in MQTT v5 - the OASIS
Committee Specification [MQTT-OASIS-Standard-v5], it is possible to
indicate failure and provide a reason code. Section 3 describes in
more detail how MQTT v5 handles PUBLISH authorization errors.

2.2.2. PUBLISH messages from the broker to the subscriber clients

To forward PUBLISH messages to the subscribing clients, the broker
identifies all the subscribers that have valid matching topic
subscriptions (i.e., the tokens are valid, and token scopes allow a
subscription to the particular topic name). The broker sends a
PUBLISH message with the topic name and the topic message to all the
valid subscribers.

In MQTT, after connection establishment, there is no way to inform a
client that an authorization error has occurred for previously
subscribed topics, e.g., token expiry. In the case of an
authorization error, the broker has two options: (1) stop forwarding
PUBLISH messages to the unauthorized client or (2) disconnect the
client. In the MQTT v3.1 - the OASIS Standard [MQTT-OASIS-Standard],
the MQTT DISCONNECT messages are only sent from a client to the
broker. Therefore, the server disconnection needs to take place
below the application layer. In MQTT v5 - the OASIS Committee
Specification [MQTT-OASIS-Standard-v5], a server-side DISCONNECT
message is possible and described in Section 3.

2.3. Authorizing SUBSCRIBE messages

In MQTT, a SUBSCRIBE message is sent from a client to the broker to
create one or more subscriptions to one or more topics. The
SUBSCRIBE message may contain multiple topic filters. The topic
filters may include wildcard characters.

On receiving the SUBSCRIBE message, the broker MUST use the type of
message (i.e., SUBSCRIBE) and the topic filter in the message header
to compare against the stored token or introspection result
(depending on the implementation, different fields of the token or
introspection result may be checked, see Section 2.1.3).

As a response to the SUBSCRIBE message, the broker issues a SUBACK
message. For each topic filter, the SUBACK packet includes a return
code matching the QoS level for the corresponding topic filter. In
the case of failure, the return code, in MQTT v3.1, must be 0x80
indicating 'Failure'. In MQTT v5, the appropriate return code is
0x87, indicating that the client is 'Not authorized'. Note that, in
both MQTT versions, a reason code is returned for each topic filter.
Therefore, the client may receive success codes for a subset of its
topic filters, while being unauthorized for the rest.

2.4. Token expiration

The broker MUST check for token expiration whenever a CONNECT,
PUBLISH or SUBSCRIBE message is received or sent. The broker SHOULD
check for token expiration on receiving a PINGREQUEST message. This
may allow for early detection of a token expiry.

The token expiration is checked by checking the 'exp' claim of a CWT/
JWT or via performing an introspection request with the Authorization
server as described in Section 8.2 of the ACE framework
[I-D.ietf-ace-oauth-authz]. In the basic operation, token
expirations MAY lead to disconnecting the associated client.
However, in MQTT v5 - the OASIS Committee Specification
[MQTT-OASIS-Standard-v5], better error handling and re-authentication
are possible. This is explained in more detail in Section 3.

2.5. Handling disconnections and retained messages

According to MQTT v3.1 - the OASIS Standard [MQTT-OASIS-Standard],
only Client DISCONNECT messages are allowed. In MQTT v5 - the OASIS
Committee Specification [MQTT-OASIS-Standard-v5], server-side
DISCONNECT messages are possible, allowing to return '0x87 Not
Authorized' return code to the client.

In the case of a DISCONNECT, due to the Clean Session flag, the
broker deletes all session state but MUST keep the retained messages.
By setting a RETAIN flag in a PUBLISH message, the publisher
indicates to the broker that it should store the most recent message
for the associated topic. Hence, the new subscribers can receive the
last sent message from the publisher for that particular topic
without waiting for the next PUBLISH message. In the case of a
disconnection, the broker MUST continue publishing the retained
messages as long as the associated tokens are valid.

In case of disconnections due to network errors or server
disconnection due to a protocol error (which includes authorization
errors), the Will message must be sent if the client supplied a Will
in the CONNECT request message. The token provided in the CONNECT
request must cover the Will topic. The Will message MUST be
published to the Will topic when the network connection is closed
regardless of whether the corresponding token has expired.

3. Improved Protocol Interactions with MQTT v5

In the new MQTT v5 - the OASIS Committee Specification Draft
[MQTT-OASIS-Standard-v5], several new capabilities are introduced,
which enable better integration with the ACE standards. The newly
enhanced authentication and re-authentication methods support a wider
range of authentication flows beyond username and password. With the
MQTT v5, there is a clearly defined approach for using token-based
approaches. Similarly, in MQTT v5, it is possible for a client to
request a re-authentication. Finally, MQTT v5 generally improves
error reporting, enabling better response to authorization failures
during publishing messages to the subscribers.

3.1. Token Transport via Authentication Exchange (AUTH)

To initiate the authentication and authorization flow, as before, the
CAS initiates the token request as in Section 2.1. When the client
wishes to connect to the RS (broker), it uses the CONNECT message of
MQTT. Figure 4 shows the structure of the MQTT CONNECT control
message used in MQTT v5.

```
      0           8          16          24          32
      +-------------------------------------------------+
      |CPT=1 | Rsvd.|Remaining len.| Protocol  name len. = 4 |
      +-------------------------------------------------+
      |                'M' 'Q' 'T' 'T'                  |
      +-------------------------------------------------+
      | Proto.level=5|Connect flags|        Keep alive      |
      +-------------------------------------------------+
      |                Property length                  |
      |          Auth. Method (0x15) | 'ace_mqtt_tls'   |
      |          Auth. Data (0x16)   | empty or token   |
      |                                                 |
      +-------------------------------------------------+
```
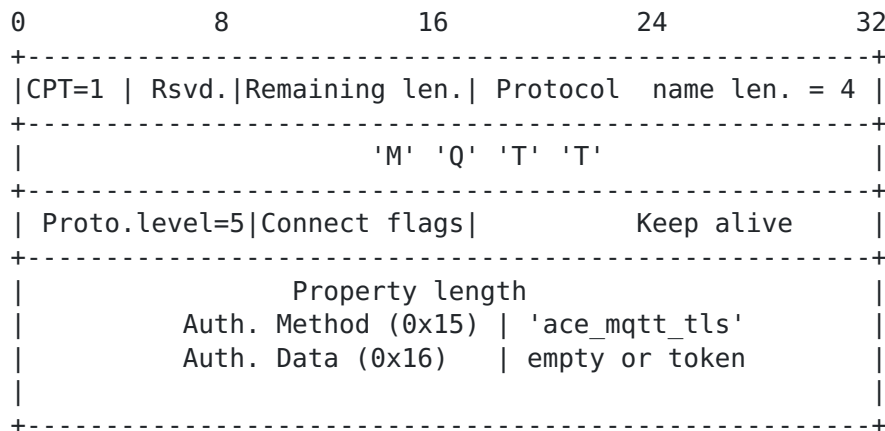
Figure 4: MQTT CONNECT control message. (CPT=Control Packet Type,
         Rsvd=Reserved, len.=length, Proto.=Protocol)

To communicate the necessary connection parameters, the client uses
the appropriate flags of the CONNECT message. To achieve a clean
session (i.e., the session should start without an existing session),
the new MQTT v5 session flags MUST be set appropriately: the Clean
Start Flag MUST be set to 1 and Session Expiry Interval MUST be set
to 0.

With the enhanced authentication capabilities, it is not necessary to
overload the username and password fields in the CONNECT message for
ACE authentication. Nevertheless, the RS MUST support both methods
for supporting the token: (1) Token transport via username and
password and (2) using the new AUTH (Authentication Exchange) method.
The token transport via username and password is as described in
Section 2.1.2. The rest of this section describes the AUTH method.

To use the AUTH method, the username flag MUST be set to 0, and the
password flag MUST be set to 0. The client can set the
Authentication Method as a property of a CONNECT packet by setting
Auth Properties (with the property identifier 0x15). The client must
MUST set the UTF-8 encoded string containing the name of the
authentication method as 'ace_mqtt_tls'. If the RS does not support
this profile, it sends a CONNACK with a Reason Code of '0x8C (Bad
authentication method)'

The Authentication Method is followed by the Authentication Data,
which has a property identifier 0x16. Authentication data is binary
data and is defined by the authentication method. The RS MAY support
different implementations for transporting the authentication data.
The first option is that Authentication data contains both the token
and the keyed message digest (MAC) or signature as described in
Section 2.1.2. The encoding of this field MAY use CBOR and COSE. In

this case, the token validation proceeds as described in
Section 2.1.3 and the server responds with a CONNACK. The reason
code of the CONNACK '0x00 (Success)' if the authentication is
successful. In case of an invalid PoP token, the CONNACK reason code
is '0x87 (Not Authorized)'.

The second option that RS may accept is a challenge/response
protocol. If the Authentication Data only includes the token, the RS
MUST respond with an AUTH packet, with the Authenticate Reason Code
set to '0x18 (Continue Authentication)'. This packet includes the
Authentication Method, which MUST be set to 'ace_mqtt_tls' and
Authentication Data. The Authentication Data MUST NOT be empty and
contains a challenge for the client. The client responds to this
with an AUTH packet, with a reason code '0x18 (Continue
Authentication)'. Similarly, the client packet sets the
Authentication Method to 'ace_mqtt_tls'. The Authentication Data in
the client's response contains the signature or MAC computed over the
RS's challenge. To this, the server responds with a CONNACK and
return code '0x00 (Success)' if the authentication is successful. In
case of an invalid PoP token, the CONNACK reason code is '0x87 (Not
Authorized)'.

Finally, this document allows the CONNECT message to have an empty
Authentication Data field. This is the AS discovery option and the
RS responds with the CONNACK reason code '0x87 (Not Authorized)' and
includes a User Property for the AS information. AS Information
contains the absolute URI of AS, and MAY also contain a nonce as
defined in the Section 5.1.2 of the ACE framework
[I-D.ietf-ace-oauth-authz]. This information MAY be CBOR encoded.

3.2. Authorization Errors and Client Re-authentication

MQTT v5 allows better error reporting. To take advantage of this for
PUBLISH messages, the QoS level should be set to greater than or
equal to 1. This guarantees that RS responds with either a PUBACK or
PUBREC packet with reason code '0x87 (Not authorized)' in the case of
an authorization error. Similarly, for the SUBSCRIBE case, the
SUBACK packet has a reason code set to '0x87 (Not authorized)' for
the unauthorized topic(s). When RS is forwarding PUBLISH messages to
the subscribed clients, it may discover that some of the subscribers
are no more authorized due to expired tokens. In this case, the RS
SHOULD send a DISCONNECT message with the reason code '0x87 (Not
authorized)'. Note that the server-side DISCONNECT is a new feature
of MQTT v5 (in MQTT v3.1 server needed to drop the connection). RS
MUST stop forwarding messages to the unauthorized subscribers.

In the case of a PUBACK with '0x87 (Not authorized)', the client can
update its token using the Re-authentication feature of MQTT v5.

Also, the clients can proactively update their tokens without waiting
for such a PUBACK. To re-authenticate, the client sends an AUTH
packet with reason code '0x19 (Re-authentication)'. The client MUST
set the authentication method as 'ace_mqtt_tls' and transport the new
token in the Authentication Data. The client and the RS go through
the same steps for proof of possession validation as described in the
previous section. If the re-authentication fails, the server MUST
send a DISCONNECT with the reason code '0x87 (Not Authorized)'.

4. IANA Considerations

The following registrations are done for the ACE OAuth Profile
Registry following the procedure specified in
[I-D.ietf-ace-oauth-authz].

Note to the RFC editor: Please replace all occurrences of "[RFC-
XXXX]" with the RFC number of this specification and delete this
paragraph.

Profile name: mqtt_tls

Profile description: Profile for delegating client authentication and
authorization using MQTT as the application protocol and TLS For
transport layer security.

Profile ID:

Change controller: IESG

Reference: [RFC-XXXX]

5. Security Considerations

This document specifies a profile for the Authentication and
Authorization for Constrained Environments (ACE) framework
[I-D.ietf-ace-oauth-authz]. Therefore, the security considerations
outlined in [I-D.ietf-ace-oauth-authz] apply to this work.

In addition, the security considerations outlined in MQTT v3.1 - the
OASIS Standard [MQTT-OASIS-Standard] and MQTT v5 - the OASIS
Committee Specification [MQTT-OASIS-Standard-v5] apply. Mainly, this
document provides an authorization solution for MQTT, the
responsibility of which is left to the specific implementation in
MQTT v5 - the OASIS Committee Specification [MQTT-OASIS-Standard-v5].
In the following, we comment on a few relevant issues based on the
current MQTT specifications.

In this document, RS uses the PoP access token to authenticate the client. If the client is able, TLS certificates sent from the client can be used by the RS to authenticate the client. The TLS certificate from the RS MUST be used by the client to authenticate the RS.

To authorize a client's publish and subscribe requests in an ongoing session, the RS caches the access token after accepting the connection from the client. However, if some permissions are revoked in the meantime, the RS may still grant publish/subscribe to revoked topics until the session ends or the token expires. When permissions change dynamically, it is expected that AS follows a reasonable expiration strategy for the access tokens.

The RS may monitor client behaviour to detect potential security problems, especially those affecting availability. These include repeated token transfer attempts to the public "authz-info" topic, repeated connection attempts, abnormal terminations, and clients that connect but do not send any data. If the RS supports the public "authz-info" topic, described in Appendix B, then this may be vulnerable to a DDoS attack, where many clients use the "authz-info" public topic to transport fictitious tokens, which RS may need to store indefinitely.

6. Privacy Considerations

The privacy considerations outlined in [I-D.ietf-ace-oauth-authz] apply to this work.

In MQTT, the RS is a central trusted party and may forward potentially sensitive information between clients. Clients may choose to encrypt the payload of their messages. However, this would not provide privacy for other properties of the message such as topic name.

7. References

7.1. Normative References

[I-D.gerdes-ace-dtls-authorize]
          Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and
          L. Seitz, "Datagram Transport Layer Security (DTLS)
          Profile for Authentication and Authorization for
          Constrained Environments (ACE)", draft-gerdes-ace-dtls-
          authorize-01 (work in progress), March 2017.

[I-D.ietf-ace-oauth-authz]
          Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and
          H. Tschofenig, "Authentication and Authorization for
          Constrained Environments (ACE) using the OAuth 2.0
          Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-15
          (work in progress), September 2018.

[MQTT-OASIS-Standard]
          Banks, A., Ed. and R. Gupta, Ed., "OASIS Standard MQTT
          Version 3.1.1 Plus Errata 01", 2015, <http://docs.oasis-
          open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>.

[MQTT-OASIS-Standard-v5]
          Banks, A., Ed., Briggs, E., Ed., Borgendale, K., Ed., and
          R. Gupta, Ed., "OASIS MQTT Version 5.0 Committee
          Specification 02", 2017, <http://docs.oasis-
          open.org/mqtt/mqtt/v5.0/csprd01/mqtt-v5.0-csprd01.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
          Requirement Levels", BCP 14, RFC 2119,
          DOI 10.17487/RFC2119, March 1997,
          <https://www.rfc-editor.org/info/rfc2119>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data
          Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006,
          <https://www.rfc-editor.org/info/rfc4648>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
          2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
          May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[fremantle14]
          Fremantle, P., Aziz, B., Kopecky, J., and P. Scott,
          "Federated Identity and Access Management for the Internet
          of Things", research International Workshop on Secure
          Internet of Things, September 2014,
          <http://dx.doi.org/10.1109/SIoT.2014.8>.

[I-D.ietf-ace-actors]
          Gerdes, S., Seitz, L., Selander, G., and C. Bormann, "An
          architecture for authorization in constrained
          environments", draft-ietf-ace-actors-06 (work in
          progress), November 2017.

[I-D.ietf-ace-cwt-proof-of-possession]
          Jones, M., Seitz, L., Selander, G., Erdtman, S., and H.
          Tschofenig, "Proof-of-Possession Key Semantics for CBOR
          Web Tokens (CWTs)", draft-ietf-ace-cwt-proof-of-
          possession-03 (work in progress), June 2018.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2",
          FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007,
          <https://www.rfc-editor.org/info/rfc4949>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework",
          RFC 6749, DOI 10.17487/RFC6749, October 2012,
          <https://www.rfc-editor.org/info/rfc6749>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-
          Possession Key Semantics for JSON Web Tokens (JWTs)",
          RFC 7800, DOI 10.17487/RFC7800, April 2016,
          <https://www.rfc-editor.org/info/rfc7800>.

Appendix A. Checklist for profile requirements

o AS discovery: For the basic protocol using either MQTT v3.1 or
   MQTT v5, the clients/client authorization servers need to be
   configured out-of-band. RS does not provide any hints to help AS
   discovery. AS discovery is possible with the MQTT v5 extensions
   described in Section 3.

o The communication protocol between the client and RS: MQTT

o The security protocol between the client and RS: TLS

o Client and RS mutual authentication: RS provides a server
   certificate during TLS handshake. Client transports token and MAC
   via the MQTT CONNECT message.

o Content format: For the HTTPS interactions with AS, "application/
   json". The MQTT payloads may be formatted JSON or CBOR.

o PoP protocols: Either symmetric or asymmetric keys can be
   supported.

o Unique profile identifier: mqtt_tls

o Token introspection: RS uses HTTPS /introspect interface of AS.

o Token request: CAS uses HTTPS /token interface of AS.

   o /authz-info endpoint: It MAY be supported using the method
      described in Appendix B, not protected.

   o Token transport: In MQTT CONNECT message or using the AUTH
      extensions for MQTT v5 described in Section 3.

Appendix B. The authorization information endpoint

   The main document described a method for transporting tokens inside
   MQTT CONNECT messages. In this section, we describe an alternative
   method to transport an access token.

   The method consists of the MQTT broker accepting PUBLISH messages to
   a public "authz-info" topic. A client using this method MUST first
   connect to the broker, and publish the access token using the "authz-
   info" topic. The broker must verify the validity of the token (i.e.,
   through local validation or introspection). After publishing the
   token, the client disconnects from the broker and is expected to try
   reconnecting over TLS.

   In MQTT v3.1, after the client published to the "authz-info" topic,
   it is not possible for the broker to communicate the result of the
   token verification. In MQTT v5, the broker can return 'Not
   authorized' error to a PUBLISH request for QoS greater or equal to 1.
   In any case, any token authorization failure affect the subsequent
   TLS handshake, which can prompt the client to obtain a valid token.

Appendix C. Document Updates

   Version 01 updates Version 00 as follows:

   o Adds Section 3 to describe improvements to the basic protocol
      operation with the new MQTT v5 - OASIS Committee Specification
      [MQTT-OASIS-Standard-v5], including improved authentication
      exchange and error reporting.

   o Condenses background information specific to MQTT in Section 2.

   o Clarifies token transport and token structure in Section 2.1.2 and
      Section 2.1.3.

   o Removes Appendix on error reporting as this is now handled with
      MQTT v5.

   Version 02 updates Version 01 as follows:

   o Adds PINGREQ packet for token expiry checks.

   o Minor typo fixes.

   Version 03 updates Version 02 as follows:

   o Requirements language fixed according to the new IETF
      recommendation.

   o The use of audience and scopes claims in access tokens has been
      clarified.

   o Encoding used in the access token has been clarified.

   o Sections on IANA, security and privacy considerations are added.

Acknowledgements

   The authors would like to thank Ludwig Seitz for his input on the
   authorization information endpoint, presented in the appendix.

Authors' Addresses

   Cigdem Sengul
   Nominet
   2 Kingdom Street
   London W2 6BD
   UK

   Email: Cigdem.Sengul@nominet.uk

   Anthony Kirby
   Nominet
   Minerva House, Edmund Halley Road
   Oxford OX4 4DQ
   UK

   Email: Anthony.Kirby@nominet.uk

   Paul Fremantle
   University of Portsmouth
   School of Computing, Buckingham House
   Portsmouth PO1 3HE
   UK

   Email: paul.fremantle@port.ac.uk