ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 21, 2021

C. Sengul
Brunel University
A. Kirby
Oxbotica
December 18, 2020

Message Queuing Telemetry Transport (MQTT)-TLS profile of Authentication and Authorization for Constrained Environments (ACE) Framework

draft-ietf-ace-mqtt-tls-profile-10

Abstract

This document specifies a profile for the ACE (Authentication and Authorization for Constrained Environments) framework to enable authorization in a Message Queuing Telemetry Transport (MQTT)-based publish-subscribe messaging system. Proof-of-possession keys, bound to OAuth2.0 access tokens, are used to authenticate and authorize MQTT Clients. The protocol relies on TLS for confidentiality and MQTT server (broker) 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 June 21, 2021.

Copyright Notice

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 Servers (Brokers) use MQTT to exchange Application Messages. The protocol relies on TLS for communication security between entities. The MQTT protocol interactions are described based on the MQTT v5.0 - the OASIS Standard [MQTT-OASIS-Standard-v5]. Since it is expected that MQTT deployments will continue to support MQTT v3.1.1 clients, this document also describes a reduced set of protocol interactions for MQTT v3.1.1 - the OASIS Standard [MQTT-OASIS-Standard]. However, MQTT v5.0 is the RECOMMENDED version as it works more naturally with ACE-style authentication and authorization.

MQTT is a publish-subscribe protocol and after connecting to the MQTT Server (Broker), a Client can publish and subscribe to multiple topics. The Broker, which acts as the Resource Server (RS), is responsible for distributing messages published by the publishers to their subscribers. In the rest of the document the terms "RS", "MQTT Server" and "Broker" are used interchangeably.

Messages are published under a Topic Name, and subscribers subscribe to the Topic Names to receive the corresponding messages. The Broker uses the Topic Name in a published message to determine which subscribers to relay the messages. In this document, topics, more specifically, Topic Names, are treated as resources. The Clients are assumed to have identified the publish/subscribe topics of interest out-of-band (topic discovery is not a feature of the MQTT protocol). A Resource Owner can pre-configure policies at the Authorisation Server (AS) that give Clients publish or subscribe permissions to different topics.

Clients prove their permission to publish and subscribe to topics hosted on an MQTT broker using an access token, bound to a proof-of-possession (PoP) key. This document describes how to authorize the following exchanges between the Clients and the Broker.

- Connection requests from the Clients to the Broker
- Publish requests from the Clients to the Broker, and from the Broker to the Clients
- Subscribe requests from Clients to the Broker

Clients use MQTT PUBLISH message to publish to a topic. This document does not protect the payload of the PUBLISH message from the Broker. Hence, the payload is not signed or encrypted specifically for the subscribers. This functionality MAY be implemented using the

proposal outlined in the ACE Pub-Sub Profile [I-D.ietf-ace-pubsub-profile].

To provide communication confidentiality and RS authentication, TLS is used, and TLS 1.3 [RFC8446] is RECOMMENDED. This document makes the same assumptions as Section 4 of the ACE framework [I-D.ietf-ace-oauth-authz] regarding Client and RS registration with the AS and setting up keying material. While the Client-Broker exchanges are only over MQTT, the required Client-AS and RS-AS interactions are described for HTTPS-based communication [RFC7230], using 'application/ace+json' content type, and unless otherwise specified, using JSON encoding. The token MAY be a reference or JSON Web Token (JWT) [RFC7519]. For JWTs, this document follows [RFC7800] for PoP semantics for JWTs. The Client-AS and RS-AS MAY also use protocols other than HTTP, e.g. Constrained Application Protocol (CoAP) [RFC7252] or MQTT. Implementations MAY also use "application/ace+cbor" content type, and CBOR encoding [RFC8949], and CBOR Web Token (CWT) [RFC8392] and associated PoP semantics to reduce the protocol memory and bandwidth requirements. For more information, see Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs) [RFC8747].

1.1. Requirements Language

The keywords "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

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

The terminology for entities in the architecture is defined in OAuth 2.0 [RFC6749] such as "Client" (C), "Resource Server" (RS) and "Authorization Server" (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. This document also defines a Client Authorisation Server, for Clients that are not able to support HTTP.

Client Authorization Server (CAS)

An entity that prepares and endorses authentication and authorization data for a Client, and communicates using HTTPS to the AS.

1.3. MQTT-Related Terminology

The document describes message exchanges as MQTT protocol interactions. The Clients are MQTT Clients, which connect to the Broker to publish and subscribe to Application Messages, labelled with their topics. For additional information, please refer to the MQTT v5.0 - the OASIS Standard [MQTT-OASIS-Standard-v5] or the MQTT v3.1.1 - the OASIS Standard [MQTT-OASIS-Standard].

MQTTS
: Secured transport profile of MQTT. MQTTS runs over TLS.

Broker
: The Server in MQTT. It acts as an intermediary between the Clients that publish Application Messages, and the Clients that made Subscriptions. The Broker acts as the Resource Server for the Clients.

Client
: A device or program that uses MQTT.

Session
: A stateful interaction between a Client and a Broker. Some Sessions last only as long as the network connection, others can span multiple network connections.

Application Message
: The data carried by the MQTT protocol. The data has an associated Quality-of-Service (QoS) level and a Topic Name.

QoS level
: The level of assurance for the delivery of an Application Message. The QoS level can be 0-2, where "0" indicates "At most once delivery", "1" "At least once delivery", and "2" "Exactly once delivery".

Property
: The last field of the Variable Header is a set of properties for several MQTT control messages (e.g. CONNECT, CONNACK) . A Property consists of an Identifier which defines its usage and data type, followed by a value. The Identifier is encoded as a Variable Byte Integer. For example, "Authentication Data" property with an Identifier 22.

Topic Name
: The label attached to an Application Message, which is matched to a Subscription.

Subscription
: A Subscription comprises a Topic Filter and a maximum QoS. A Subscription is associated with a single session.

Topic Filter
: An expression that indicates interest in one or more Topic Names. Topic Filters may include wildcards.

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 QoS level 2.

CONNECT
: Client request to connect to the Broker. This is the first packet sent by a Client.

CONNACK
: The Broker connection acknowledgment. CONNACK packets contain return codes indicating either a success or an error state in response to a Client's CONNECT packet.

AUTH
: Authentication Exchange. An AUTH control packet is sent from the Client to the Broker or from the Broker to the Client as part of an extended authentication exchange. AUTH Properties include Authentication Method and Authentication Data. The Authentication Method is set in the CONNECT packet, and consequent AUTH packets follow the same Authentication Method. The contents of the Authentication Data are defined by the Authentication Method.

PUBLISH
: Publish request sent from a publishing Client to the Broker, or from the Broker to a subscribing Client.

PUBACK
: Response to a PUBLISH request with QoS level 1. A PUBACK can be sent from the Broker to a Client or from a Client to the Broker.

PUBREC

Response to PUBLISH request with QoS level 2. PUBREC can be sent from the Broker to a Client or from a Client to the Broker.

SUBSCRIBE
: Subscribe request sent from a Client.

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 also alive. The "Keep Alive" period is set in the CONNECT message.

PINGRESP
: Response sent by the Broker to the Client in response to PINGREQ. It indicates the Broker is alive.

Will
: If the network connection is not closed normally, the Broker sends a last Will message for the Client, if the Client provided one in its CONNECT message. If the Will Flag is set in the CONNECT flags, then the payload of the CONNECT message includes information about the Will. The information consists of the Will Properties, Will Topic, and Will Payload fields.

2. Authorizing Connection Requests

This section specifies how Client connections are authorized by the MQTT Broker. Figure 1 shows the basic protocol flow during connection set-up. The token request and response use the token endpoint at the AS, specified in Section 5.6 of the ACE framework [I-D.ietf-ace-oauth-authz]. Steps (D) and (E) are optional and use the introspection endpoint, specified in Section 5.7 of the ACE framework. The Client and the Broker use HTTPS to communicate to AS via these endpoints. The Client and the Broker use MQTT to communicate between them. The C-AS and Broker-AS communication MAY be implemented using protocols other than HTTPS, e.g. CoAP or MQTT.

If the Client is resource-constrained or does not support HTTPS, a separate Client Authorisation Server may carry out the token request on behalf of the Client, and later, onboard the Client with the token. The interactions between a Client and its Client Authorization Server for token onboarding, and support for MQTTS-based token requests at the AS are out of scope of this document.

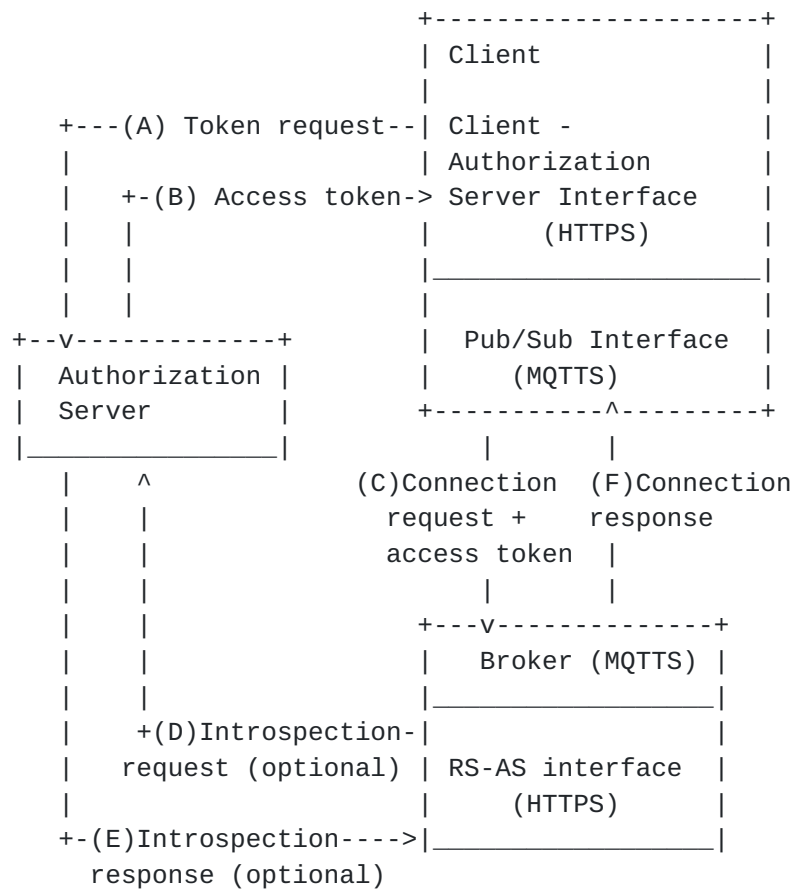

Figure 1: Connection set-up

2.1. Client Token Request to the Authorization Server (AS)

The first step in the protocol flow (Figure 1 (A)) is the token acquisition by the Client from the AS. The Client and the AS MUST perform mutual authentication. The Client requests an access token from the AS as described in Section 5.6.1 of the ACE framework [I-D.ietf-ace-oauth-authz]. The media format is 'application/ace+json'. The AS uses JSON in the payload of its responses to the Client and the RS.

If the AS successfully verifies the access token request and authorizes the Client for the indicated audience (i.e. RS) and scopes (i.e. publish/subscribe permissions over topics as described in Section 3), the AS issues an access token (Figure 1 (B)). The response includes the parameters described in Section 5.6.2 of the ACE framework [I-D.ietf-ace-oauth-authz], and specifically, the "ace_profile" parameter is set to "mqtt_tls". The returned token is a Proof-of-Possession (PoP) token by default. This document follows [RFC7800] for PoP semantics for JWTs. The PoP token includes a 'cnf' parameter with a symmetric or asymmetric PoP key. Note that the

'cnf' parameter in the web tokens are to be consumed by the RS and not the Client. For the asymmetric case, the PoP token may include the 'rs_cnf' parameter containing the information about the public key to be used by the RS to authenticate as described in [I-D.ietf-ace-oauth-params].

The AS returns error responses for JSON-based interactions following Section 5.2 of [RFC6749]. When CBOR is used, the interactions MUST implement Section 5.6.3 of the ACE framework [I-D.ietf-ace-oauth-authz].

2.2. Client Connection Request to the Broker (C)

2.2.1. Client-Server Authentication over TLS and MQTT

The Client and the Broker MUST perform mutual authentication. The Client MUST authenticate to the Broker either over MQTT or TLS. For MQTT, the options are "None" and "ace". For TLS, the options are "Anon" for an anonymous client, and "Known(RPK/PSK)" for Raw Public Keys (RPK) [RFC7250] and Pre-Shared Keys (PSK), respectively. Combined, client authentication has the following options:

- "TLS:Anon-MQTT:None": This option is used only for the topics that do not require authorization, including the "authz-info" topic. Publishing to the "authz-info" topic is described in Section 2.2.2.

- "TLS:Anon-MQTT:ace": The token is transported inside the CONNECT message, and MUST be validated using one of the methods described in Section 2.2.2. This option also supports a tokenless connection request for AS discovery.

- "TLS:Known(RPK/PSK)-MQTT:none": For the RPK, the token MUST have been published to the "authz-info" topic. For the PSK, the token MAY be, alternatively, provided as an "identity" in the "identities" field in the client's "pre_shared_key" extension in TLS 1.3. The TLS session set-up is as described in DTLS profile for ACE [I-D.ietf-ace-dtls-authorize].

- "TLS:Known(RPK/PSK)-MQTT:ace": This option SHOULD NOT be chosen as the token transported in the CONNECT overwrites any permissions passed during the TLS authentication.

It is RECOMMENDED that the Client implements "TLS:Anon-MQTT:ace" as a first choice when working with protected topics. However, depending on the Client capability, Client MAY implement "TLS:Known(RPK/PSK)-MQTT:none", and consequently "TLS:Anon-MQTT:None" to submit its token to "authz-info".

The Broker MUST support "TLS:Anon-MQTT:ace". To support Clients with different capabilities, the Broker MAY provide multiple client authentication options, e.g. support "TLS:Known(RPK)-MQTT:none" and "TLS:Anon-MQTT:None", to enable RPK-based client authentication, but fall back to "TLS:Anon-MQTT:ace" if the Client does not send a client certificate (i.e. it sends an empty Certificate message) during the TLS handshake.

The Broker MUST be authenticated during the TLS handshake. If the Client authentication uses TLS:Known(RPK/PSK), then the Broker is authenticated using the respective method. Otherwise, to authenticate the Broker, the client MUST validate a public key from a X.509 certificate or an RPK from the Broker against the 'rs_cnf' parameter in the token response. The AS MAY include the thumbprint of the RS's X.509 certificate in the 'rs_cnf' (thumbprint as defined in [I-D.ietf-cose-x509]). In this case, the client MUST validate the RS certificate against this thumbprint.

2.2.2. authz-info: The Authorization Information Topic

In the cases when the Client MUST transport the token to the Broker first, the Client connects to the Broker to publish its token to the "authz-info" topic. The "authz-info" topic MUST be publish-only (i.e. the Clients are not allowed to subscribe to it). "authz-info" is not protected, and hence, the Client uses the "TLS:Anon-MQTT:None" option over a TLS connection. After publishing the token, the Client disconnects from the Broker and is expected to reconnect using client authentication over TLS (i.e. TLS:Known(RPK/PSK)-MQTT:none).

The Broker stores and indexes all tokens received to the "authz-info" topic in its key store (similar to DTLS profile for ACE [I-D.ietf-ace-dtls-authorize]). This profile follows the recommendation of Section 5.8.1 of the ACE framework [I-D.ietf-ace-oauth-authz], and expects that the Broker stores only one token per proof-of-possession key, and any other token linked to the same key overwrites an existing token.

The Broker MUST verify the validity of the token (i.e. through local validation or introspection, if the token is a reference) as described in Section 2.2.5. If the token is not valid, the Broker MUST discard the token. Depending on the QoS level of the PUBLISH message, the Broker returns the error response as a PUBACK or a DISCONNECT message as explained below.

If the QoS level is equal to 0, and the token is invalid or the claims cannot be obtained in the case of an introspected token, the Broker MUST send a DISCONNECT message with the reason code '0x87 (Not authorized)'. If the PUBLISH payload does not parse to a token, the

RS MUST send a DISCONNECT with the reason code '0x99 (Payload format invalid)'.

If the QoS level of the PUBLISH message is greater than or equal to 1, the Broker MUST return 'Not authorized' in PUBACK. If the PUBLISH payload does not parse to a token, the PUBACK reason code is '0x99 (Payload format invalid)'.

It must be noted that when the RS sends the 'Not authorized' response, this corresponds to the token being invalid, and not that the actual PUBLISH message was not authorized. Given that the "authz-info" is a public topic, this response is not expected to cause confusion.

2.2.3. Transporting Access Token Inside the MQTT CONNECT

This section describes how the Client transports the token to the Broker (RS) inside the CONNECT message. If this method is used, the Client TLS connection is expected to be anonymous, and the Broker is authenticated during the TLS connection set-up. The approach described in this section is similar to an earlier proposal by Fremantle et al [fremantle14].

After sending the CONNECT, the client MUST wait to receive the CONNACK from the Broker. The only messages it is allowed to send are DISCONNECT or AUTH that is in response to the Broker AUTH. Similarly, the Broker MUST NOT process any packets before it has sent a CONNACK. The only exceptions are DISCONNECT or an AUTH response from the Client.

Figure 2 shows the structure of the MQTT CONNECT message used in MQTT v5.0. A CONNECT message is composed of a fixed header, a variable header and a payload. The fixed header contains the Control Packet Type (CPT), Reserved, and Remaining Length fields. The Variable Header contains the Protocol Name, Protocol Level, Connect Flags, Keep Alive, and Properties fields. The Connect Flags in the variable header specify the properties of the MQTT session. It also indicates the presence or absence of some fields in the Payload. The payload contains one or more encoded fields, namely a unique Client identifier for the Client, a Will Topic, Will Payload, User Name and Password. All but the Client identifier can be omitted depending on the flags in the Variable Header.

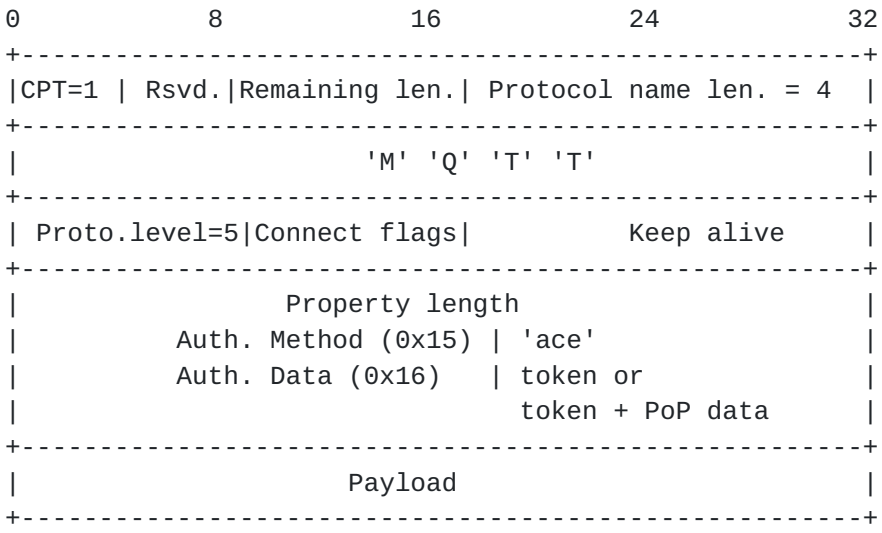

Figure 2: MQTT v5 CONNECT control message with ACE authentication. (CPT=Control Packet Type)

The CONNECT message flags are Username, Password, Will retain, Will QoS, Will Flag, Clean Start, and Reserved. Figure 3 shows how the flags MUST be set to use AUTH packets for authentication and authorisation, i.e. the username and password flags MUST be set to 0. An MQTT v5.0 RS MAY also support token transport using Username and Password to provide a security option for MQTT v3.1.1 clients, as described in Section 6.

User name Flag	Pass. Flag	Will retain	Will QoS	Will Flag	Clean Start	Rsvd.
0	0	X	X X	X	X	0

Figure 3: CONNECT flags for AUTH

The Will Flag indicates that a Will message needs to be sent if the network connection is not closed normally. The situations in which the Will message is published include disconnections due to I/O or network failures, and the server closing the network 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 stores 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. To avoid publishing Will Messages in the case of temporary network disconnections, the Client specifies a Will Delay

Interval in the Will Properties. Section 5 explains how the Broker deals with the retained messages in further detail.

In MQTT v5.0, the Client signals a clean session (i.e. the session does not continue an existing session), by setting the Clean Start Flag to 1 and, the Session Expiry Interval to 0 in the CONNECT message. In this profile, the Broker SHOULD always start with a clean session regardless of how these parameters are set. Starting a clean session helps the Broker avoid keeping unnecessary session state for unauthorised clients. If the Broker starts a clean session, the Broker MUST set the Session Present flag to 0 in the CONNACK packet to signal this to the Client.

The Broker MAY support session continuation e.g., if the Broker requires it for QoS reasons. With session continuation, the Broker maintains and uses client state from the existing session. The session state kept at the server MAY include token and its introspection result (for reference tokens) in addition to the MQTT session state. The MQTT session state is identified by the Client identifier and includes state on client subscriptions; messages with QoS levels 1 and 2, and which have not been completely acknowledged or pending transmission to the Client; and if the Session is currently not connected, the time at which the Session will end and Session State will be discarded.

When reconnecting to a Broker that supports session continuation, the Client MUST still provide a token, in addition to using the same Client identifier, setting the Clean Start to 0 and supplying a Session Expiry interval in the CONNECT message. The Broker MUST perform proof-of-possession validation on the provided token. If the token matches the stored state, the Broker MAY skip introspecting a token by reference, and use the stored introspection result. The Broker MUST also verify the Client is authorized to receive or send packets that are pending transmission. When a Client connects with a long Session Expiry Interval, the Broker may need to maintain Client's MQTT session state after it disconnects for an extended period. Brokers SHOULD implement administrative policies to limit misuse.

Note that, according to the MQTT standard, the Broker uses the Client identifier to identify the session state. In the case of a Client identifier collision, a client may take over another client's session. Given that clients provide a token at each connection, clients will only send or receive messages to their authorized topics. Therefore, while this issue is not expected to affect security, it may affect QoS (i.e. PUBLISH or QoS messages saved for Client A may be delivered to a Client B). In addition, if this Client identifier represents a Client already connected to the

Broker, the Broker sends a DISCONNECT packet to the existing Client with Reason Code of '0x8E (Session taken over)', and closes the connection to the client.

2.2.4. Authentication Using AUTH Property

To use AUTH, the Client MUST set the Authentication Method as a property of a CONNECT packet by using the property identifier 21 (0x15). This is followed by a UTF-8 Encoded String containing the name of the Authentication Method, which MUST be set to 'ace'. 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 22 (0x16) and is binary data. The binary data in MQTT is represented by a two-byte integer length, which indicates the number of data bytes, followed by that number of bytes. Based on the Authentication Data, RS MUST support both options below:

- Proof-of-Possession using a challenge from the TLS session

- Proof-of-Possession via Broker generated challenge/response

2.2.4.1. Proof-of-Possession Using a Challenge from the TLS session

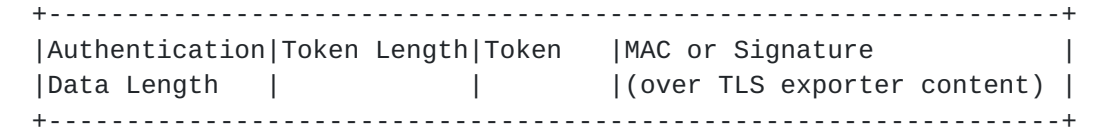

Figure 4: Authentication Data for PoP based on TLS exporter content

For this option, the Authentication Data MUST contain the two-byte integer token length, the token, and the keyed message digest (MAC) or the Client signature (as shown in Figure 4). The Proof-of-Possession key in the token is used to calculate the keyed message digest (MAC) or the Client signature based on the content obtained from the TLS exporter ([RFC5705] for TLS 1.2 and for TLS 1.3, Section 7.5 of [RFC8446]). This content is exported from the TLS session using the exporter label 'EXPORTER-ACE-MQTT-Sign-Challenge', an empty context, and length of 32 bytes. The token is also validated as described in Section 2.2.5 and the server responds with a CONNACK with the appropriate response code. The client cannot reauthenticate using this method during the same session (see Section 4).

2.2.4.2. Proof-of-Possession via Broker-generated Challenge/Response

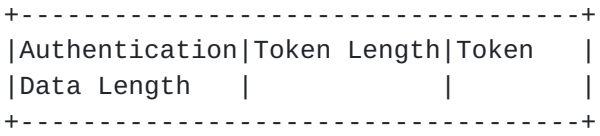

Figure 5: Authentication Data to Initiate PoP based on Challenge/ Response

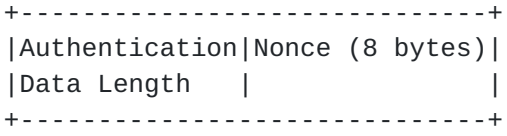

Figure 6: Authentication Data for Broker Challenge

For this option, the RS follows a Broker-generated challenge/response protocol. If the Authentication Data contains only the two-byte integer token length and the token (as shown in Figure 5), 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" and Authentication Data. The Authentication Data MUST NOT be empty and contains an 8-byte nonce as a challenge for the Client (Figure 6).

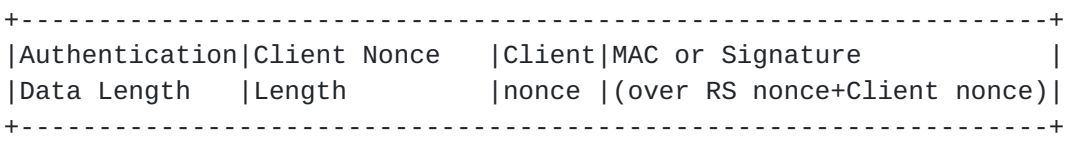

Figure 7: Authentication Data for Client Challenge Response

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". The Authentication Data in the Client's response is formatted as shown in Figure 7 and includes the client nonce length, the client nonce, and the signature or MAC computed over the RS nonce concatenated with the client nonce using PoP key in the token.

Next, the token is validated as described in Section 2.2.5. The success case is illustrated in Figure 8. The client MAY also re-authenticate using this challenge-response flow, as described in Section 4.

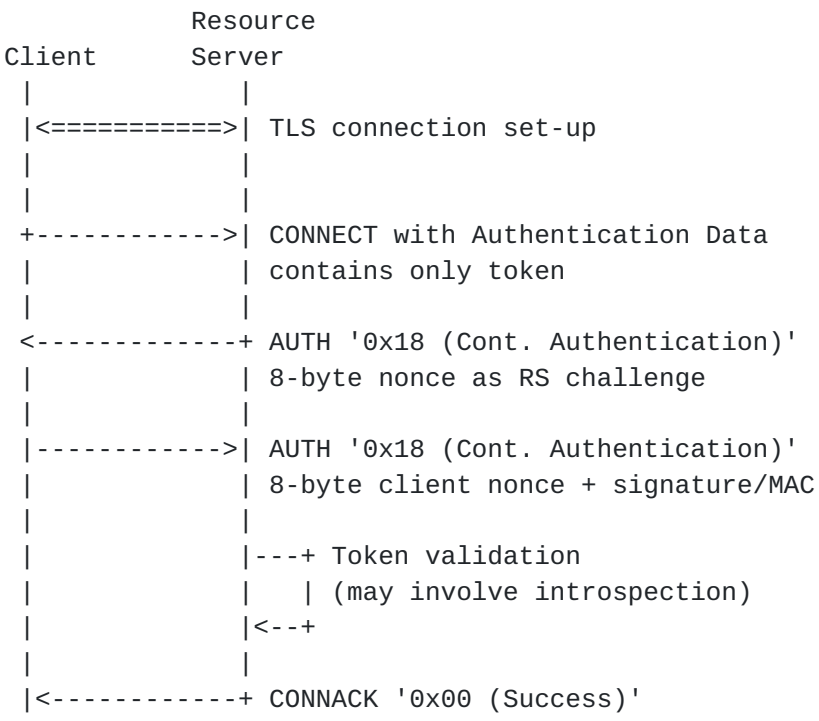

Figure 8: PoP Challenge/Response Flow - Success

2.2.5. Token Validation

The RS MUST verify the validity of the token either locally (e.g. in the case of a self-contained token) or the RS MAY send an introspection request to the AS. The RS MUST verify the claims according to the rules set in the Section 5.8.1.1 of the ACE framework [I-D.ietf-ace-oauth-authz].

To authenticate the Client, the RS validates the signature or the MAC, depending on how the PoP protocol is implemented. HS256 (HMAC-SHA-256) [RFC6234] and Ed25519 [RFC8032] are mandatory to implement depending on the choice of symmetric or asymmetric validation. Validation of the signature or MAC MUST fail if the signature algorithm is set to "none", when the key used for the signature algorithm cannot be determined, or the computed and received signature/MAC do not match.

2.2.6. 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.

2.2.6.1. Unauthorised Request and the Optional Authorisation Server Discovery

If the Client does not provide a valid token or omits the Authentication Data field, or the token or Authentication data are malformed, authentication fails. The Broker responds with the CONNACK reason code "0x87 (Not Authorized)"

The Broker MAY also trigger AS discovery, and include a User Property (identified by 38 (0x26)) in the CONNACK for the AS Request Creation Hints. The User Property is a UTF-8 string pair, composed of a name and a value. The name of the User Property MUST be set to "ace_as_hint". The value of the user property is a UTF-8 encoded JSON string containing the mandatory "AS" parameter, and the optional parameters "audience", "kid", "cnonce", and "scope" as defined in Section 5.1.2 of the ACE framework [I-D.ietf-ace-oauth-authz].

2.2.6.2. Authorisation Success

On success, the reason code of the CONNACK is "0x00 (Success)". The AS informs the client that selected profile is "mqtt_tls" using the "ace_profile" parameter in the token response. If the Broker starts a new session, it MUST also set Session Present to 0 in the CONNACK packet to signal a clean session to the Client. Otherwise, it MUST set Session Present to 1.

If the Broker accepts the connection, it MUST store the token until the end of the connection. On Client or Broker disconnection, the Client is expected to transport a token again on the next connection attempt.

If the token is not self-contained and the Broker uses token introspection, it MAY cache the validation result to authorize the subsequent PUBLISH and SUBSCRIBE messages. PUBLISH and SUBSCRIBE messages, which are sent after a connection 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. The Broker SHOULD also use a cache time out to introspect tokens regularly.

3. Authorizing PUBLISH and SUBSCRIBE Messages

To authorize a Client's PUBLISH and SUBSCRIBE messages, the Broker uses the scope field in the token (or in the introspection result). The scope field contains the publish and subscribe permissions for the Client. The scope is a JSON array, each item following the Authorization Information Format (AIF) for ACE [I-D.ietf-ace-aif]. Using the Concise Data Definition Language (CDDL) [RFC8610], the specific data model for MQTT is:

```
AIF-MQTT = AIF-Generic<topic_filter, permissions>
AIF-Generic<topic_filter, permissions> = [*[topic_filter, permissions]]
topic_filter = tstr
permissions = [+permission]
permission = "pub"/"sub"
```

Figure 9: AIF-MQTT data model

Topic filters are implemented according to Section 4.7 of MQTT v5.0 - the OASIS Standard [MQTT-OASIS-Standard-v5] and includes special wildcard characters. The multi-level wildcard, '#', matches any number of levels within a topic, and the single-level wildcard, '+', matches one topic level.

If the scope is empty i.e., the JSON array is empty, the RS records no permissions for the client for any topic. In this case, the client is not able to publish or subscribe to any protected topics.

An example scope may contain:

```
[["topic1", ["pub","sub"]], ["topic2/#",["pub"]], ["+/topic3",["sub"]]]
```

Figure 10: Example scope

This access token gives publish ("pub") and subscribe ("sub") permissions to the "topic1", publish permission to all the subtopics of "topic2", and subscribe permission to all "topic3", skipping one level.

If the Will Flag is set, then the Broker MUST check that the token allows the publication of the Will message (i.e. the Will Topic filter is in the scope array).

3.1. 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 match against the scope array items in the cached token or its introspection result. Following the example in the previous section, a client sending a PUBLISH message to 'topic2/a' would be allowed, as the scope array includes the '["topic2/#",["pub"]]'.

If the Client is allowed to publish to the topic, the Broker publishes the message to all valid subscribers of the topic. In the case of an authorization failure, the Broker MUST return an error, if the Client has set the QoS level of the PUBLISH message to greater than or equal to 1. Depending on the QoS level, the Broker responds with either a PUBACK or PUBREC packet with reason code '0x87 (Not

authorized)'. On receiving an acknowledgement with '0x87 (Not authorized)', the Client MAY reauthenticate by providing a new token as described in Section 4.

For QoS level 0, the Broker sends a DISCONNECT with reason code "0x87 (Not authorized)" and closes the network connection. Note that the server-side DISCONNECT is a new feature of MQTT v5.0 (in MQTT v3.1.1, the server needs to drop the connection).

3.2. 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). The Broker sends a PUBLISH message with the Topic name to all the valid subscribers.

The Broker MUST NOT forward messages to the unauthorized subscribers. There is no way to inform the Clients with invalid tokens that an authorization error has occurred other than sending a DISCONNECT message. The Broker SHOULD send a DISCONNECT message with the reason code '0x87 (Not authorized)'.

3.3. 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 match against the scope field of the stored token or introspection result. The Topic Filters MUST be equal or a subset of at least one of the 'topic_filter' fields in the scope array found in the Client's token.

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 is 0x87, indicating that the Client is 'Not authorized'. 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.

4. Token Expiration, Update and Reauthentication

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. The Broker MAY also check for token expiration periodically, e.g. every hour. This may allow for early detection of a token expiry.

The token expiration is checked by checking the 'exp' claim of a JWT or introspection response, or via performing an introspection request with the AS as described in Section 5.7 of the ACE framework [I-D.ietf-ace-oauth-authz]. Token expirations may trigger the RS to send PUBACK, SUBACK and DISCONNECT messages with return code set to "Not authorized". After sending a DISCONNECT message, the network connection is closed, and no more messages can be sent.

If the Client used the challenge-respose PoP as defined in Section 2.2.4.2, the Client MAY reauthenticate as a response to the PUBACK and SUBACK that signal loss of authorization. The Clients MAY also proactively update their tokens, i.e. before they receive a message with a "Not authorized" return code. To start reauthentication, the Client MUST send an AUTH packet with the reason code "0x19 (Re-authentication)". The Client MUST set the Authentication Method as "ace" and transport the new token in the Authentication Data. The Broker accepts reauthentication requests if the Client has already submitted a token (may be expired) and validated via the challenge-response PoP. Otherwise, the Broker MUST deny the request. If the reauthentication fails, the Broker MUST send a DISCONNECT with the reason code "0x87 (Not Authorized)".

5. Handling Disconnections and Retained Messages

In the case of a Client DISCONNECT, the Broker deletes all the session state but MUST keep the retained messages. By setting a RETAIN flag in a PUBLISH message, the publisher indicates to the Broker to 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. 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 is sent if the Client supplied a Will in the CONNECT message. The Client's token scope array MUST include the Will Topic. The Will message MUST be published to the Will Topic regardless of whether the corresponding token has expired. In the

case of a server-side DISCONNECT, the server returns the '0x87 Not Authorized' return code to the Client.

6. Reduced Protocol Interactions for MQTT v3.1.1

This section describes a reduced set of protocol interactions for the MQTT v3.1.1 Clients. An MQTT v5.0 Broker MAY implement these interactions for the MQTT v3.1.1 clients; MQTT v5.0 clients are NOT RECOMMENDED to use the flows described in this section. Brokers that do not support MQTT v3.1.1 clients return a CONNACK packet with Reason Code '0x84 (Unsupported Protocol Version)' in response to the connection requests.

6.1. Token Transport

As in MQTT v5.0, the token MAY either be transported before by publishing to the "authz-info" topic, or inside the CONNECT message.

In MQTT v3.1.1, after the Client published to the "authz-info" topic, the Broker cannot communicate the result of the token validation as PUBACK reason codes or server-side DISCONNECT messages are not supported. In any case, an invalid token would fail the subsequent TLS handshake, which can prompt the Client to obtain a valid token.

To transport the token to the Broker inside the CONNECT message, the Client uses the username and password fields. Figure 11 shows the structure of the MQTT CONNECT message.

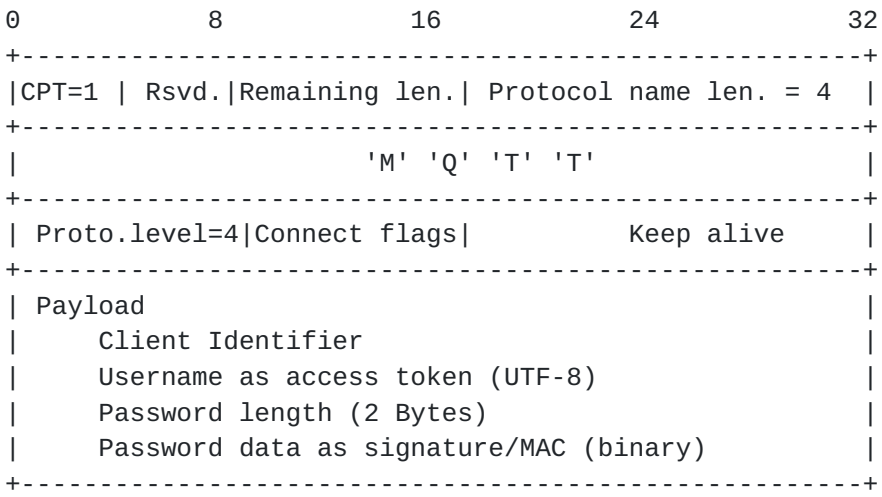

Figure 11: MQTT CONNECT control message. (CPT=Control Packet Type, Rsvd=Reserved, len.=length, Proto.=Protocol)

Figure 12 shows how the MQTT connect flags MUST be set to initiate a connection with the Broker.

User name flag	Pass. flag	Will retain	Will QoS	Will Flag	Clean	Rsvd.
1	1	X	X X	X	X	0

Figure 12: MQTT CONNECT flags. (Rsvd=Reserved)

The Broker SHOULD NOT accept session continuation. To this end, the Broker ignores how the Clean Session Flag is set, and on connection success, the Broker MUST set the Session Present flag to 0 in the CONNACK packet to indicate a clean session to the Client. If the Broker wishes to support session continuation, it MUST still perform proof-of-possession validation on the provided Client token. MQTT v3.1.1 does not use a Session Expiry Interval, and the Client expects that the Broker maintains the session state after it disconnects. However, stored Session state can be discarded as a result of administrator policies, and Brokers SHOULD implement the necessary policies to limit misuse.

The Client MAY set the Will Flag as desired (marked as 'X' in Figure 12). 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 in MQTT v3.1.1 does not have a field to indicate the authentication method. To signal that the Username field contains an ACE token, this field MUST be prefixed with 'ace' keyword, which is followed by 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 MUST apply the PoP key on the challenge derived from the TLS session as described in Section 2.2.4.1.

In MQTT v3.1.1, the MQTT Username is a UTF-8 encoded string (i.e. is prefixed by a 2-byte length field followed by UTF-8 encoded character data) and may be up to 65535 bytes. Therefore, an access token that is not a valid UTF-8 MUST be Base64 [RFC4648] encoded. (The MQTT Password allows binary data up to 65535 bytes.)

6.2. Handling Authorization Errors

Handling errors are more primitive in MQTT v3.1.1 due to not having appropriate error fields, error codes, and server-side DISCONNECTs. Therefore, the broker will disconnect on almost any error and may not keep session state, necessitating clients to make a greater effort to ensure that tokens remain valid and not attempt to publish to topics

that they do not have permissions for. The following lists how the broker responds to specific errors.

- CONNECT without a token: It is not possible to support AS discovery via sending a tokenless CONNECT message to the Broker. This is because a CONNACK packet in MQTT v3.1.1 does not include a means to provide additional information to the Client. Therefore, AS discovery needs to take place out-of-band. The tokenless CONNECT attempt MUST fail.

- Client-RS PUBLISH authorization failure: In the case of a failure, it is not possible to return an error in MQTT v3.1.1. 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, as DISCONNECT messages are only sent from a Client to the Broker, the server disconnection needs to take place below the application layer.

- SUBSCRIBE authorization failure: In the SUBACK packet, the return code is 0x80 indicating 'Failure' for the unauthorized topic(s). Note that, in both MQTT versions, a reason code is returned for each Topic Filter.

- RS-Client PUBLISH authorization failure: 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. These token expirations SHOULD lead to disconnecting the Client rather than silently dropping messages.

7. IANA Considerations

This document registers 'EXPORTER-ACE-MQTT-Sign-Challenge' (introduced in Section 2.2.4.1 in this document) in the TLS Exporter Label Registry [RFC8447].

In addition, 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.

Name: mqtt_tls

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

CBOR Value:

Reference: [RFC-XXXX]

8. 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 v5.0 - the OASIS Standard [MQTT-OASIS-Standard-v5] and MQTT v3.1.1 - the OASIS Standard [MQTT-OASIS-Standard] apply. Mainly, this document provides an authorization solution for MQTT, the responsibility of which is left to the specific implementation in the MQTT standards. In the following, we comment on a few relevant issues based on the current MQTT specifications.

After the RS validates an access token and accepts a connection from a client, it caches the token to authorize a Client's publish and subscribe requests in an ongoing session. RS does not cache any invalid tokens. If a client's permissions get revoked but the access token has not expired, the RS may still grant publish/subscribe to revoked topics. If the RS caches the token introspection responses, then the RS SHOULD use a reasonable cache timeout to introspect tokens regularly. When permissions change dynamically, it is expected that AS also 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 Section 2.2.2, 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.

For MQTT v5.0, when a Client connects with a long Session Expiry Interval, the RS may need to maintain Client's MQTT session state after it disconnects for an extended period. For MQTT v3.1.1, the session state may need to be stored indefinitely, as it does not have

a Session Expiry Interval feature. The RS SHOULD implement administrative policies to limit misuse of the session continuation by the Client.

9. 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. This document does not protect the contents of the PUBLISH message from the Broker, and hence, the content of the PUBLISH message is not signed or encrypted separately for the subscribers. This functionality may be implemented using the proposal outlined in the ACE Pub-Sub Profile [I-D.ietf-ace-pubsub-profile]. However, this solution would still not provide privacy for other properties of the message such as Topic Name.

10. References

10.1. Normative References

[I-D.ietf-ace-aif]
Bormann, C., "An Authorization Information Format (AIF) for ACE", draft-ietf-ace-aif-00 (work in progress), July 2020.

[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-36 (work in progress), November 2020.

[I-D.ietf-ace-oauth-params]
Seitz, L., "Additional OAuth Parameters for Authorization in Constrained Environments (ACE)", draft-ietf-ace-oauth-params-13 (work in progress), April 2020.

[I-D.ietf-cose-x509]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates", draft-ietf-cose-x509-08 (work in progress), December 2020.

[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 Standard MQTT Version 5.0", 2017, <http://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.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>.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <https://www.rfc-editor.org/info/rfc5705>.

[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>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/info/rfc8447>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

[RFC8747] Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", RFC 8747, DOI 10.17487/RFC8747, March 2020, <https://www.rfc-editor.org/info/rfc8747>.

10.2. 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-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-ietf-ace-dtls-authorize-14 (work in progress), October 2020.

[I-D.ietf-ace-pubsub-profile] Palombini, F., "Pub-Sub Profile for Authentication and Authorization for Constrained Environments (ACE)", draft-ietf-ace-pubsub-profile-01 (work in progress), July 2020.

[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>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/info/rfc8392>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

Appendix A. Checklist for profile requirements

- AS discovery: AS discovery is possible with the MQTT v5.0 described in Section 2.2.
- The communication protocol between the Client and RS: MQTT
- The security protocol between the Client and RS: TLS
- Client and RS mutual authentication: Several options are possible and described in Section 2.2.1.
- Content format: For the HTTPS interactions with AS, "application/ace+json".
- PoP protocols: Either symmetric or asymmetric keys can be supported.
- Unique profile identifier: mqtt_tls
- Token introspection: RS uses HTTPS /introspect interface of AS.
- Token request: Client or its Client AS uses HTTPS /token interface of AS.

- /authz-info endpoint: It MAY be supported using the method described in Section 2.2.2, but is not protected.

- Token transport: Via "authz-info" topic, or in MQTT CONNECT message for both versions of MQTT. AUTH extensions also used for authentication and re-authentication for MQTT v5.0 as described in Section 2.2 and in Section 4.

Appendix B. Document Updates

Version 07 to 08:

- Fixed several nits, typos based on WG reviews.

- Added missing references.

- Added the definition for Property defined by MQTT, and Client Authorisation Server.

- Added artwork to show Authorisation Data format for various PoP-related message exchange.

- Removed all MQTT-related must/should/may.

- Made AS discovery optional.

- Clarified what the client and server must implement for client authentication; cleaned up TLS 1.3 related language.

Version 06 to 07:

- Corrected the title.

- In Section 2.2.3, added the constraint on which packets the Client can send, and the server can process after CONNECT before CONNACK.

- In Section 2.2.3, clarified that session state is identified by Client Identifier, and listed its content.

- In Section 2.2.3, clarified the issue of Client Identifier collision, when the broker supports session continuation.

- Corrected the buggy scope example in Section 3.1.

Version 05 to 06:

- Replace the originally proposed scope format with AIF model. Defined the AIF-MQTT, gave an example with a JSON array. Added a normative reference to the AIF draft.

- Clarified client connection after submitting token via "authz-info" topic as TLS:Known(RPK/PSK)-MQTT:none.

- Expanded acronyms on their first use including the ones in the title.

- Added a definition for "Session".

- Corrected "CONNACK" definition, which earlier said it's the first packet sent by the broker.

- Added a statement that the the broker will disconnect on almost any error and may not keep session state.

- Clarified that the broker does not cache invalid tokens.

Version 04 to 05:

- Reorganised Section 2 such that "Unauthorised Request: Authorisation Server Discovery" is presented under Section 2.

- Fixed Figure 2 to remove the "empty" word.

- Clarified that MQTT v5.0 Brokers may implement username/password option for transporting the ACE token only for MQTT v.3.1.1 clients. This option is not recommended for MQTT v.5.0 clients.

- Changed Clean Session requirement both for MQTT v.5.0 and v.3.1.1. The Broker SHOULD NOT, instead of MUST NOT, continue sessions. Clarified expected behaviour if session continuation is supported. Added to the Security Considerations the potential misuse of session continuation.

- Fixed the Authentication Data to include token length for the Challenge/Response PoP.

- Added that Authorisation Server Discovery is triggered if a token is invalid and not only missing.

- Clarified that the Broker should not accept any other packets from Client after CONNECT and before sending CONNACK.

- Added that client reauthentication is accepted only for the challenge/response PoP.

- Added Ed25519 as mandatory to implement.

- Fixed typos.

Version 03 to 04:

- Linked the terms Broker and MQTT server more at the introduction of the document.

- Clarified support for MQTTv3.1.1 and removed phrases that might be considered as MQTTv5 is backwards compatible with MQTTv3.1.1

- Corrected the Informative and Normative references.

- For AS discovery, clarified the CONNECT message omits the Authentication Data field. Specified the User Property MUST be set to "ace_as_hint" for AS Request Creation Hints.

- Added that MQTT v5 brokers MAY also implement reduced interactions described for MQTTv3.1.1.

- Added to Section 3.1, in case of an authorisation failure and QoS level 0, the RS sends a DISCONNECT with reason code '0x87 (Not authorized)'.

- Added a pointer to section 4.7 of MQTTv5 spec for more information on topic names and filters.

- Added HS256 and RSA256 are mandatory to implement depending on the choice of symmetric or asymmetric validation.

- Added MQTT to the TLS exporter label to make it application specific: 'EXPORTER-ACE-MQTT-Sign-Challenge'.

- Added a format for Authentication Data so that length values prefix the token (or client nonce) when Authentication Data contains more than one piece of information.

- Clarified clients still connect over TLS (server-side) for the authz-info flow.

Version 02 to 03:

- Added the option of Broker certificate thumbprint in the 'rs_cnf' sent to the Client.

- Clarified the use of a random nonce from the TLS Exporter for PoP, added to the IANA requirements that the label should be registered.

- Added a client nonce, when Challenge/Response Authentication is used between Client and Broker.

- Clarified the use of the "authz-info" topic and the error response if token validation fails.

- Added clarification on wildcard use in scopes for publish/subscribe permissions

- Reorganised sections so that token authorisation for publish/subscribe messages are better placed.

Version 01 to 02:

- Clarified protection of Application Message payload as out of scope, and cited draft-palombini-ace-coap-pubsub-profile for a potential solution

- Expanded Client connection authorization to capture different options for Client and Broker authentication over TLS and MQTT

- Removed Payload (and specifically Client Identifier) from proof-of-possession in favor of using tls-exporter for a TLS-session based challenge.

- Moved token transport via "authz-info" topic from the Appendix to the main text.

- Clarified Will scope.

- Added MQTT AUTH to terminology.

- Typo fixes, and simplification of figures.

Version 00 to 01:

- Present the MQTTv5 as the RECOMMENDED version, and MQTT v3.1.1 for backward compatibility.

- Clarified Will message.

- Improved consistency in the use of terminology and upper/lower case.

- Defined Broker and MQTTS.
- Clarified HTTPS use for C-AS and RS-AS communication. Removed reference to actors document, and clarified the use of client authorization server.
- Clarified the Connect message payload and Client Identifier.
- Presented different methods for passing the token and PoP.
- Added new figures to explain AUTH packets exchange, updated CONNECT message figure.

Acknowledgements

The authors would like to thank Ludwig Seitz for his review and his input on the authorization information endpoint. The authors would like to thank Paul Fremantle for the initial discussions on MQTT v5.0 support.

Authors' Addresses

Cigdem Sengul
Brunel University
Dept. of Computer Science
Uxbridge UB8 3PH
UK

Email: csengul@acm.org

Anthony Kirby
Oxbotica
1a Milford House, Mayfield Road, Summertown
Oxford OX2 7EL
UK

Email: anthony@anthony.org