ACE Working Group C. Sengul
Internet-Draft A. Kirby
Intended status: Standards Track Nominet
Expires: July 28, 2017 January 24, 2017

MQTT-TLS profile of ACE
draft-sengul-kirby-ace-mqtt-tls-profile-00

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 publishing and subscribing 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 http://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 July 28, 2017.

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. Protocol interactions follow MQTT v3.1 - OASIS Standard [MQTT-OASIS-Standard]. Future releases may enable improvements to the protocol operation (e.g., by allowing MQTT message return codes for authorization errors).

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 the purpose of 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. Both publisher and subscriber clients use an access token, each 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. In order to provide communication confidentiality and resource server authentication, TLS is used.

The publisher and subscriber 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] in terms of client and RS registration with the AS and establishing of keying material.

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

- Connection establishment between the clients and the broker
- Publish messages from the publishers to the broker, and from the broker to the subscribers
- Subscribe messages from the subscribers to the broker

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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", which is defined in Section 1.2. 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 - OASIS Standard [MQTT-OASIS-Standard].

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 of 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.

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 broker to a client is a CONNACK packet. CONNACK packets contain return codes indicating either a success or an error state to the client.

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

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

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

SUBSCRIBE
: The client subscribe request.

SUBACK
: Subscribe acknowledgment.

2. Protocol Interactions

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

- Authorizing connection establishment between the clients and the broker
- Authorizing publish messages from the publishers to the broker, and from the broker to the subscribers
- 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 the ACE framework [I-D.ietf-ace-oauth-authz].

Figure 1 shows the basic protocol flow during connection establishment.

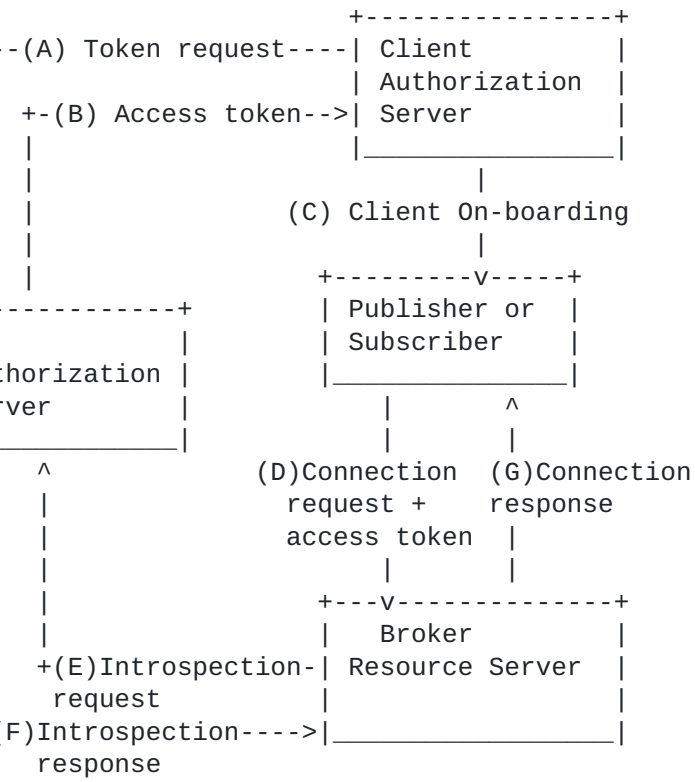

Figure 1: Connection establishment

2.1.1. Client Authorization Server (CAS) to Authorization Server (AS)

The first step in the protocol flow is 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 MQTT, 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 response contains a token and a 'cnf' parameter with a symmetric or asymmetric proof-of-possession (PoP) key.

The token request is similar to the examples presented in Section 6.1. of the ACE framework [I-D.ietf-ace-oauth-authz] with a modified profile name 'mqtt_tls'.

2.1.2. Authorization Server (AS) to Client Authorization Server (CAS)

If the access token request has been successfully verified by the AS and the client is authorized to obtain a PoP token for the indicated audience (i.e., broker) and scopes (i.e., publish/subscribe to the requested topics), the AS issues an access token. The response includes the parameters described in Section 6.2 of the ACE framework [I-D.ietf-ace-oauth-authz].

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].

2.1.3. Client connection request to the broker

Having received the token, the client can use it to request an MQTT connection to the broker over a TLS session with server authentication. This document 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 Freemantle et al. [freemantle14]. 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 allow clients to publish to "authz-info" topic unauthorized, and in this case, "authz-info" topic SHOULD be publish only (i.e., the clients are not allowed to subscribe to it). Implementation of the public "authz-info" topic is discussed 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.

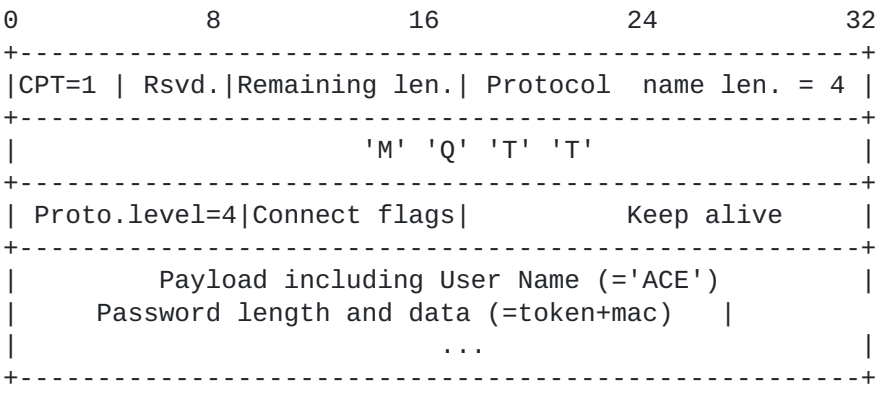

Figure 2: MQTT CONNECT control message. (CPT=Control Packet Type, Rsvd=Reserved, len.=length, Proto.=Protocol)

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

Figure 3: MQTT CONNECT flags. (Rsvd=Reserved)

In order 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. The Will QoS specifies the QoS level of the Will message, and may be set to 0x00, 0x01 or 0x02. The Will retain flag may be set as desired.

If it is set to 1, the Will message will be delivered to all future subscribers whose subscriptions match the Will topic. Section 2.6 explains how the broker deals with the RETAINed messages in further detail.

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

The Username and Password field are used to indicate to the resource server that the CONNECT message is carrying an ACE token and the MAC of the request. To this end, the Username is set to 'ACE', and the Password field is populated with a JSON object containing the token and the MAC. The Password length field MUST be set to the size of the JSON object. (The maximum size of the password field is defined as 65535 bytes by MQTT v3.1 - OASIS Standard [MQTT-OASIS-Standard].)

Figure 4 shows an example for setting the password field in an MQTT CONNECT message.

```
{
"token": b64'SLAV32hkKG....,
"mac": b64'kDZvddkndxvhGRXZhvuDjEWhGeE=,
}
```

Figure 4: Example token and MAC as password data in CONNECT message.

2.1.4. Token validation

RS MUST verify the validity of the token. This validation MAY be done locally 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 'expires_in' parameter 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 'mac' parameter in the password field to prove the possession of the key. The

resource server validates the 'mac' over 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. Scope strings MAY follow an application specific convention e.g., 'publish_topic1' or 'subscribe_topic2'. If the Will flag is set, then the broker MUST check that the token allows the publication of the Will message too.

The broker MAY cache the introspection result, because it will need to decide whether to accept subsequent PUBLISH and SUBSCRIBE messages, and these messages, which are sent after a connection is set-up, may not contain tokens. If the introspection result is not cached, then the RS needs to introspect the saved token for each request.

2.1.5. 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 authorization server), the broker MUST send a CONNACK message to the client.

The following responses may be returned to the client.

- If the broker accepts the connection, the broker MUST send a CONNACK message with Return Code 0x00 indicating 'Connection Accepted'.

- If the connection is denied, the broker MUST send a CONNACK message with 0x05 indicating 'Connection Refused, not authorized'.

- If the data in the user name or password is malformed, the broker MUST send a CONNACK message with 0x04 indicating 'Connection Refused, bad user name or password'.

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 have a payload. Therefore, AS discovery is assumed to have taken place out-of-band.

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

2.2. Authorizing PUBLISH messages

Figure 5 shows the PUBLISH message used in MQTT, which includes fixed

and variable headers.

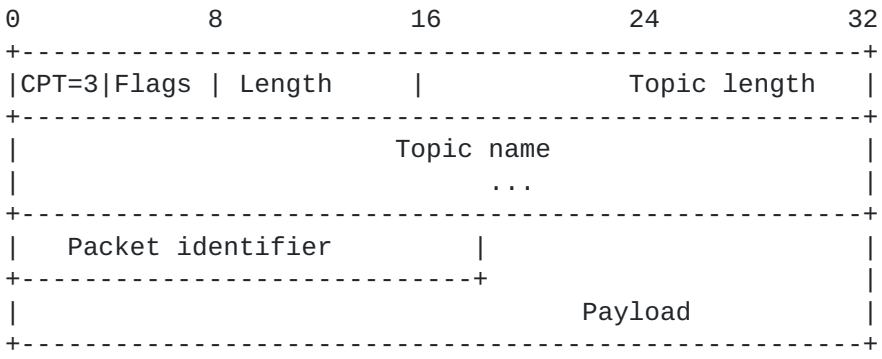

Figure 5: MQTT PUBLISH control message. (CPT=Control Packet Type)

The variable header includes flags for QoS and RETAIN. If RETAIN is set to 1, then the broker must store the most recent application message per topic, and its QoS, to forward to future subscribers. Other fields in the PUBLISH header are topic name and packet identifier. The topic name MUST NOT include wildcard characters according to MQTT v3.1 - OASIS Standard [MQTT-OASIS-Standard].

2.2.1. PUBLISH messages from the publisher client to the broker

The payload of PUBLISH messages contains an application message. The content and the format of the data is application specific. Therefore, the client MAY include its token inside the PUBLISH messages. The token could for example be included as:

```
{
"message":"topic message",
"token": b64'SLAV32hkKG....,
}
```

Figure 6: Example token in PUBLISH payload.

If the application message contains a token, the broker MAY locally inspect the token or MAY use the /introspect interface of the authorization server. The token received in the PUBLISH message MAY be different than the one stored after connection handshake. On receiving a new token, the RS discards any previously stored token for the client and MUST store the new token as not all PUBLISH messages may carry tokens.

If the application message does not contain a token, the broker MUST use the type of message (i.e., PUBLISH) and the topic name in the message header to compare against the 'scope' field of the cached introspection result for the client.

If the client is allowed to publish to the topic, the broker may return an acknowledgment message. This is determined by the QoS flags in the Flags field of the PUBLISH header. If QoS level is 0, the RS does not send back a response. If the QoS level is equal or greater than 1, the RS must respond with an acknowledgement message (i.e., PUBACK for QoS=1 or PUBREC for QoS=2). These messages can currently only indicate success, and there is no equivalent 'NACK' to indicate failure. Next, the RS must publish the message to all valid subscribers to the topic.

In the case of an authorization error, the broker SHOULD disconnect the client. Otherwise, it MUST silently ignore the message. In the MQTT v3.1 - OASIS Standard [MQTT-OASIS-Standard], the MQTT DISCONNECT messages are only sent from a client to the broker. So, server disconnection needs to take place below the application layer. Appendix C describes an alternative method for handling authorization errors, possibly avoiding disconnections.

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 matching valid 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 - 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. Appendix C describes an alternative method, where disconnections may be avoided.

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.

Figure 7 shows the MQTT SUBSCRIBE message format. The SUBSCRIBE message may contain multiple topic filters. The topic filters may include wildcard characters.

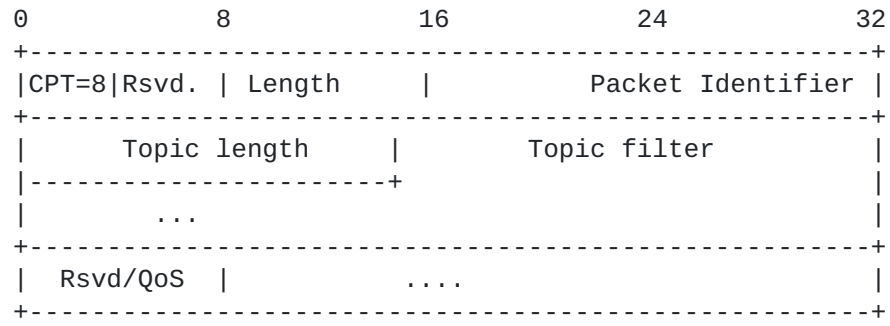

Figure 7: MQTT SUBSCRIBE control message. (CPT=Control Packet Type, Rsvd.=Reserved, QoS=Quality of Service)

The SUBSCRIBE message does not have any field suitable for including a token. Therefore, on receiving the SUBSCRIBE message, the broker MUST use the type of message (i.e., SUBSCRIBE) and the topic name in the message header to compare against the 'scope' field of the stored token or introspection result.

As a response to the SUBSCRIBE message, the broker issues a SUBACK message. For each topic filter, the SUBACK packet includes a return code. In the case of success, the return code must be either 0x00, 0x01 or 0x02, matching the QoS level for the corresponding topic filter.

In the case of failure, the return code must be 0x08 indicating 'Failure'. There is no other way to signal the reason for an authorization failure to the Subscriber, or to communicate further detail. Appendix C describes an alternative method where more detailed error messages can be provided to the client.

2.4. Token expiration

The broker checks for token expiration whenever a CONNECT, PUBLISH or SUBSCRIBE message is received or sent. The validation is done either by checking the 'exp' claim of a CWT/JWT or via performing an introspection request with the Authorization server as described in the Section 8.2 of the ACE framework [I-D.ietf-ace-oauth-authz]. Token expirations leads to disconnecting the associated client. Appendix C describes an alternative method, where clients are allowed to update tokens, avoiding disconnections.

2.5. Handling disconnections

According to MQTT v3.1 - OASIS Standard [MQTT-OASIS-Standard], only Client DISCONNECT messages are allowed. (This is expected to change in the future, enabling Server DISCONNECT messages.) In the case of

a Client DISCONNECT, due to the Clean Session flag, the broker deletes all session state but MUST keep the retained messages and send them according to methodology described in Section 2.6. 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 the authorization errors), the Will message must be sent if the client supplied a Will in the CONNECT request message (see Figure 3). According to the [MQTT-OASIS-Standard], if the CONNECT request is accepted, then any WILL message must be stored. The Will message must be published to the Will topic when the network connection is closed.

2.6. Handling retained messages

The broker treats retained messages according to the [MQTT-OASIS-Standard]. By setting a RETAIN flag in a PUBLISH message (or in a CONNECT message for the Will message), the publisher indicates to the broker that it should store the most recent message for the associated topic, so the broker can send the message to any future subscribers. Hence, the new subscribers can receive the last sent message from the publisher for that particular topic, without waiting for the next PUBLISH message.

According to the [MQTT-OASIS-Standard], if a publisher client disconnects, the retained messages do not form part of the session state and must not be deleted by the broker (since the Clean session flag set to 1 during the connection request, othersession state is deleted). Therefore, the broker MUST continue publishing retained messages for as long as the stored token for the client is valid: This applies to both the PUBLISH and WILL messages. However, if the disconnection is triggered by the broker due to an authorisation error, the broker MUST stop publishing all retained messages from that client.

3. IANA Considerations

This memo includes no request to IANA.

4. Security Considerations

TBD.

5. Privacy Considerations

TBD.

6. References

6.1. 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-00 (work in progress), October 2016.

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE)", draft-ietf-ace-oauth-authz-04 (work in progress), October 2016.

[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>.

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

6.2. Informative References

[freemantle14]
Freemantle, 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-04 (work in progress), September 2016.

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

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

Appendix A. Checklist for profile requirements

- AS discovery: The clients/client authorization servers need to be configured out-of-band. RS does not provide any hints to help AS discovery.
- Communication protocol between the client and RS: MQTT
- Security protocol between the client and RS: TLS
- Client and RS mutual authentication: RS provides a server certificate during TLS handshake. Client uses token and MAC fields in the MQTT connect message.
- Content format: For the HTTPS interactions with AS, "application/json". The MQTT payloads may be formatted JSON or CBOR.
- 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: CAS uses HTTPS /token interface of AS.
- /authz-info endpoint: It MAY be supported using the method described in Appendix B, not protected.
- Token transport: In MQTT CONNECT message or using the method described in Appendix B.

Appendix B. The `authorization information' endpoint

The main document described a method where the access token is transported inside the MQTT CONNECT message. In this section, we describe an alternative method to transport the access token.

The method consists of the MQTT broker providing 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.

After the client published to the "authz-info" topic, it is not possible for the broker to communicate the result of the verification. The response to a PUBLISH message may be a PUBACK or PUBREC, and these messages indicate successful reception of the PUBLISH message and cannot communicate authorization errors. However, the token failure will affect the TLS handshake, which may be used to prompt the client to obtain a valid token. In Appendix C, an alternative method for error handling is discussed.

Appendix C. Error handling and token updates

Section 2.1.3 uses the CONNECT message to transfer the PoP token to the broker. This is simple, with only two states: 'Disconnected' and 'Authorized' (see Figure 8). However, the result of an authorization error is a server side disconnection without any feedback or error message.

	State	
	0	1
Events	Disconnected	Authorized
CONNECT success	1	-
CONNECT failure	0	-
Authorization expires	-	0
other disconnection		

Figure 8: Client state machine - simple token transport

To enable token updates during the lifetime of a connection, and also to allow the broker to send error messages to a client, this section proposes using a client-specific "authz-info-${ClientId}" topic. This case requires three states: 'Disconnected', 'Connected' and 'Authorized', shown in Figure 9.

In the Disconnected state, as before, the client needs to transport the token and attempt to establish a connection. Token transport MAY be done using any of the methods mentioned in this document. The CONNECT payload SHOULD include a unique client identifier: Although in MQTT a broker may accept 0-byte client identifiers, in this use

case the client would not be aware of its client identifier, so would
be unable to update its token, or subscribe to the "authz-info-${ClientId}" topic to receive error messages.

If the CONNECT succeeds, then the client moves to the Authorized state. It SHOULD also subscribe to the topic "authz-info-${ClientId}" to be able to receive authorization errors. The subscription MUST fail if the topic name does not contain the ClientId established during the CONNECT handshake.

If the token validation fails,the broker MUST publish an authorization error to "authz-info-${ClientId}" and the client moves to the Connected state. In this state, the client MAY publish a new token or it MAY disconnect. If a new token is published, then the broker MUST verify the token and send an authorization response to "authz-info-${ClientId}" indicating success or failure. In case of success, the client moves to the "Authorized" state. In the case of failure, the client remains in "Connected" state but will not be able to publish or receive message from its subscribed topics due to authorisation problems. It MAY be able to publish/subscribe to public topics.

	State		
	0	1	2
Events	Disconnected	Connected	Authorized
CONNECT success	2	-	-
CONNECT failure	0	-	-
PUBLISH new token success	-	2	2
PUBLISH new token failure	-	1	1
Authorization expires	-	-	1
Other disconnection	-	0	0

Figure 9: Client state machine - with error handling and token update

While this solution allows better authorization error feedback, it is a more complex solution. The broker needs to maintain separate authz-info topics for separate clients.

Acknowledgements

The authors would like to thank Ludwig Seitz for his input on the authorisation information endpoint, error handling and token updates presented in the appendices.

Authors' Addresses

Cigdem Sengul
Nominet
1 Sekforde Street
London EC1R 0BE
UK

Email: Cigdem.Sengul@nominet.uk

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

Email: Anthony.Kirby@nominet.uk