

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 26 June 2022

F. Palombini
Ericsson AB
M. Tiloca
RISE AB
23 December 2021

Key Provisioning for Group Communication using ACE
draft-ietf-ace-key-groupcomm-15

Abstract

This document defines how to use the Authentication and Authorization for Constrained Environments (ACE) framework to distribute keying material and configuration parameters for secure group communication. Candidate group members acting as Clients and authorized to join a group can do so by interacting with a Key Distribution Center (KDC) acting as Resource Server, from which they obtain the keying material to communicate with other group members. While defining general message formats as well as the interface and operations available at the KDC, this document supports different approaches and protocols for secure group communication. Therefore, details are delegated to separate application profiles of this document, as specialized instances that target a particular group communication approach and define how communications in the group are protected. Compliance requirements for such application profiles are also specified.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at <https://github.com/ace-wg/ace-key-groupcomm>.

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 26 June 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Terminology	5
2. Overview	7
3. Authorization to Join a Group	10
3.1. Authorization Request	11
3.2. Authorization Response	13
3.3. Token Transferring	15
3.3.1. 'sign_info' Parameter	17
3.3.2. 'kdcchallenge' Parameter	19
4. KDC Functionalities	19
4.1. Interface at the KDC	20
4.1.1. Operations Supported by Clients	23
4.1.2. Error Handling	24
4.2. /ace-group	26
4.2.1. FETCH Handler	26
4.2.1.1. Retrieve Group Names	27
4.3. /ace-group/GROUPNAME	28
4.3.1. POST Handler	28
4.3.1.1. Join the Group	41
4.3.2. GET Handler	43
4.3.2.1. Retrieve Group Keying Material	44
4.4. /ace-group/GROUPNAME/pub-key	45
4.4.1. FETCH Handler	45
4.4.1.1. Retrieve a Subset of Public Keys in the Group . .	47
4.4.2. GET Handler	48
4.4.2.1. Retrieve All Public Keys in the Group	48
4.5. ace-group/GROUPNAME/kdc-pub-key	49
4.5.1. GET Handler	49
4.5.1.1. Retrieve the KDC's Public Key	50
4.6. /ace-group/GROUPNAME/policies	51

4.6.1.	GET Handler	51
4.6.1.1.	Retrieve the Group Policies	52
4.7.	/ace-group/GROUPNAME/num	53
4.7.1.	GET Handler	53
4.7.1.1.	Retrieve the Keying Material Version	54
4.8.	/ace-group/GROUPNAME/nodes/NODENAME	54
4.8.1.	GET Handler	55
4.8.1.1.	Retrieve Group and Individual Keying Material	56
4.8.2.	PUT Handler	57
4.8.2.1.	Request to Change Individual Keying Material	59
4.8.3.	DELETE Handler	60
4.8.3.1.	Leave the Group	60
4.9.	/ace-group/GROUPNAME/nodes/NODENAME/pub-key	61
4.9.1.	POST Handler	61
4.9.1.1.	Uploading a New Public Key	62
5.	Removal of a Group Member	63
6.	Group Rekeying Process	65
6.1.	Point-to-Point Group Rekeying	66
6.2.	One-to-Many Group Rekeying	67
6.2.1.	Protection of Rekeying Messages	69
7.	Extended Scope Format	72
8.	ACE Groupcomm Parameters	74
9.	ACE Groupcomm Error Identifiers	77
10.	Security Considerations	79
10.1.	Secure Communication in the Group	79
10.2.	Update of Group Keying Material	80
10.2.1.	Misalignment of Group Keying Material	82
10.3.	Block-Wise Considerations	83
11.	IANA Considerations	83
11.1.	Media Type Registrations	83
11.2.	CoAP Content-Formats	84
11.3.	OAuth Parameters	84
11.4.	OAuth Parameters CBOR Mappings	85
11.5.	Interface Description (if=) Link Target Attribute Values	85
11.6.	CBOR Tags	86
11.7.	ACE Groupcomm Parameters	86
11.8.	ACE Groupcomm Key Types	87
11.9.	ACE Groupcomm Profiles	87
11.10.	ACE Groupcomm Policies	88
11.11.	Sequence Number Synchronization Methods	89
11.12.	ACE Scope Semantics	89
11.13.	ACE Groupcomm Errors	90
11.14.	ACE Groupcomm Rekeying Schemes	90
11.15.	Expert Review Instructions	91
12.	References	92
12.1.	Normative References	92
12.2.	Informative References	94

Appendix A. Requirements on Application Profiles	96
A.1. Mandatory-to-Address Requirements	96
A.2. Optional-to-Address Requirements	99
Appendix B. Extensibility for Future COSE Algorithms	100
B.1. Format of 'sign_info_entry'	100
Appendix C. Document Updates	101
C.1. Version -14 to -15	101
C.2. Version -13 to -14	101
C.3. Version -05 to -13	102
C.4. Version -04 to -05	102
C.5. Version -03 to -04	103
C.6. Version -02 to -03	103
C.7. Version -01 to -02	104
C.8. Version -00 to -01	105
Acknowledgments	105
Authors' Addresses	106

1. Introduction

This document builds on the Authentication and Authorization for Constrained Environments (ACE) framework and defines how to request, distribute and renew keying material and configuration parameters to protect message exchanges in a group communication environment.

Candidate group members acting as Clients and authorized to join a group can interact with the Key Distribution Center (KDC) acting as Resource Server and responsible for that group, in order to obtain the necessary keying material and parameters to communicate with other group members.

In particular, this document defines the operations and interface available at the KDC, as well as general message formats for the interactions between Clients and KDC. At the same time, communications in the group can rely on different approaches, e.g., based on multicast [I-D.ietf-core-groupcomm-bis] or on publish-subscribe messaging [I-D.ietf-core-coap-pubsub], and can be protected in different ways.

Therefore, this document delegates details on the communication and security approaches used in a group to separate application profiles. These are specialized instances of this document, targeting a particular group communication approach and defining how communications in the group are protected, as well as the specific keying material and configuration parameters provided to group members. In order to ensure consistency and aid the development of such application profiles, this document defines a number of related compliance requirements (see Appendix A).

If the application requires backward and forward security, new keying material is generated and distributed to the group upon membership changes (rekeying). A group rekeying scheme performs the actual distribution of the new keying material, by rekeying the current group members when a new Client joins the group, and the remaining group members when a Client leaves the group. This can rely on different approaches, including efficient group rekeying schemes such as [RFC2093], [RFC2094] and [RFC2627].

Consistently with what is recommended in the ACE framework, this document uses CBOR [RFC8949] for data encoding. However, using JSON [RFC8259] instead of CBOR is possible, by relying on the conversion method specified in Sections 6.1 and 6.2 of [RFC8949].

1.1. Terminology

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

Readers are expected to be familiar with:

- * The terms and concepts described in the ACE framework [I-D.ietf-ace-oauth-authz] and in the Authorization Information Format (AIF) [I-D.ietf-ace-aif] to express authorization information. The terminology for entities in the considered architecture is defined in OAuth 2.0 [RFC6749]. In particular, this includes Client (C), Resource Server (RS), and Authorization Server (AS).
- * The terms and concepts described in CoAP [RFC7252]. Unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol".
- * The terms and concepts described in CBOR [RFC8949] and COSE [I-D.ietf-cose-rfc8152bis-struct] [I-D.ietf-cose-rfc8152bis-algs] [I-D.ietf-cose-countersign].

A principal interested to participate in group communication as well as already participating as a group member is interchangeably denoted as "Client" or "node".

Furthermore, this document uses "names" or "identifiers" for groups and nodes. Their different meanings are summarized below.

- * **Group:** a set of nodes that share common keying material and security parameters used to protect their communications with one another. That is, the term refers to a "security group".

This is not to be confused with an "application group", which has relevance at the application level and whose members share a common pool of resources or content. Examples of application groups are the set of all nodes deployed in a same physical room, or the set of nodes registered to a pub-sub topic.

The same security group might be associated to multiple application groups. Also, the same application group can be associated to multiple security groups. Further details and considerations on the mapping between the two types of group are out of the scope of this document.

- * **Key Distribution Center (KDC):** the entity responsible for managing one or multiple groups, with particular reference to the group membership and the keying material to use for protecting group communications.
- * **Group name:** the invariant once established identifier of a group. It is used in the interactions between Client, AS and RS to identify a group. A group name is always unique among the group names of the existing groups under the same KDC.
- * **GROUPNAME:** the invariant once established text string used in URIs. GROUPNAME uniquely maps to the group name of a group, although they do not necessarily coincide.
- * **Group identifier:** the identifier of the group keying material used in a group. Unlike group name and GROUPNAME, this identifier changes over time, when the group keying material is updated.
- * **Node name:** the invariant once established identifier of a node. It is used in the interactions between Client and RS and to identify a member of a group. Within the same group, a node name is always unique among the node names of all the current members of that group.
- * **NODENAME:** the invariant once established text string used in URIs to identify a member a group. Its value coincides with the node name of the associated group member.

This document additionally uses the following terminology:

- * Transport profile, to indicate a profile of ACE as per Section 5.8.4.3 of [I-D.ietf-ace-oauth-authz]. A transport profile specifies the communication protocol and communication security protocol between an ACE Client and Resource Server, as well as proof-of-possession methods, if it supports proof-of-possession access tokens, etc. Transport profiles of ACE include, for instance, [I-D.ietf-ace-oscore-profile], [I-D.ietf-ace-dtls-authorize] and [I-D.ietf-ace-mqtt-tls-profile].
- * Application profile, that defines how applications enforce and use supporting security services they require. These services may include, for instance, provisioning, revocation and distribution of keying material. An application profile may define specific procedures and message formats.

2. Overview

The full procedure can be separated in two phases: the first one follows the ACE Framework, between Client, AS and KDC; the second one is the key distribution between Client and KDC. After the two phases are completed, the Client is able to participate in the group communication, via a Dispatcher entity.

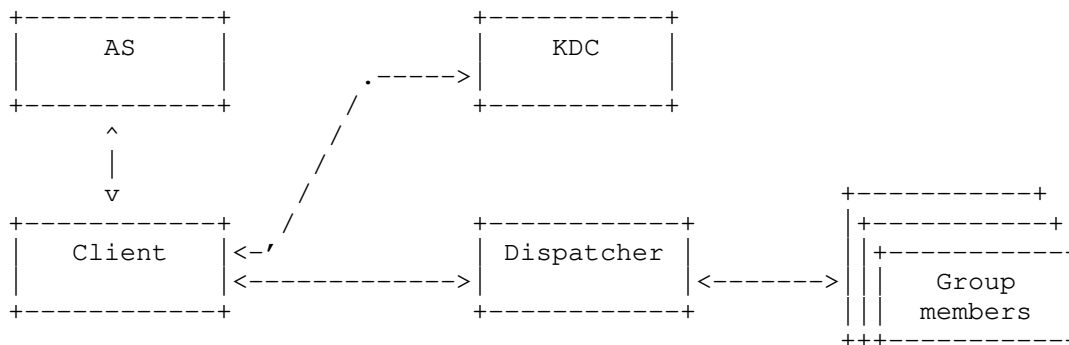


Figure 1: Key Distribution Participants

The following participants (see Figure 1) take part in the authorization and key distribution.

- * Client (C): node that wants to join a group and take part in group communication with other group members. Within the group, the Client can have different roles.
- * Authorization Server (AS): as per the AS defined in the ACE Framework, it enforces access policies, and knows if a node is allowed to join a given group with write and/or read rights.

- * Key Distribution Center (KDC): maintains the keying material to protect group communications, and provides it to Clients authorized to join a given group. During the first part of the exchange (Section 3), it takes the role of the RS in the ACE Framework. During the second part (Section 4), which is not based on the ACE Framework, it distributes the keying material. In addition, it provides the latest keying material to group members when requested or, if required by the application, when membership changes.
- * Dispatcher: entity through which the Clients communicate with the group, when sending a message intended to multiple group members. That is, the Dispatcher distributes such a one-to-many message to the group members as intended recipients. A single-recipient message intended to only one group member may be delivered by alternative means, with no assistance from the Dispatcher.

Examples of a Dispatcher are: the Broker in a pub-sub setting; a relay for group communication that delivers group messages as multiple unicast messages to all group members; an implicit entity as in a multicast communication setting, where messages are transmitted to a multicast IP address and delivered on the transport channel.

This document specifies a mechanism for:

- * Authorizing a Client to join the group (Section 3), and providing it with the group keying material to communicate with the other group members (Section 4).
- * Allowing a group member to retrieve group keying material (Section 4.8.1.1 and Section 4.8.2.1).
- * Allowing a group member to retrieve public keys of other group members (Section 4.4.1.1) and to provide an updated public key (Section 4.9.1.1).
- * Allowing a group member to leave the group (Section 5).
- * Evicting a group member from the group (Section 5).
- * Renewing and re-distributing the group keying material (rekeying) upon a membership change in the group (Section 4.8.3.1 and Section 5).

Figure 2 provides a high level overview of the message flow for a node joining a group. The message flow can be expanded as follows.

1. The joining node requests an access token from the AS, in order to access one or more group-membership resources at the KDC and hence join the associated groups.

This exchange between Client and AS MUST be secured, as specified by the transport profile of ACE used between Client and KDC. Based on the response from the AS, the joining node will establish or continue using a secure communication association with the KDC.

2. The joining node transfers authentication and authorization information to the KDC, by transferring the obtained access token. This is typically achieved by including the access token in a request sent to the /authz-info endpoint at the KDC.

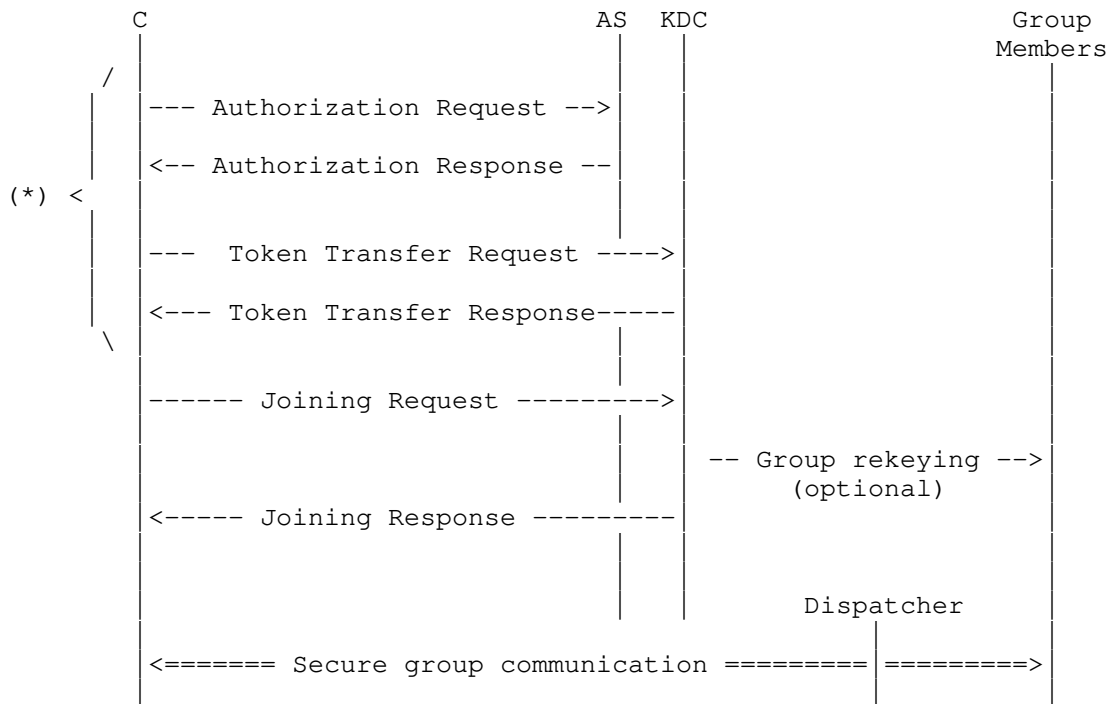
Once this exchange is completed, the joining node MUST have a secure communication association established with the KDC, before joining a group under that KDC.

This exchange and the following secure communications between the Client and the KDC MUST occur in accordance with the transport profile of ACE used between Client and KDC, such as the DTLS transport profile [I-D.ietf-ace-dtls-authorize] and OSCORE transport profile [I-D.ietf-ace-oscore-profile] of ACE.

3. The joining node starts the joining process to become a member of the group, by sending a request to the related group-membership resource at the KDC. Based on the application requirements and policies, the KDC may perform a group rekeying, by generating new group keying material and distributing it to the current group members through the rekeying scheme used in the group.

At the end of the joining process, the joining node has received from the KDC the parameters and group keying material to securely communicate with the other group members. Also, the KDC has stored the association between the authorization information from the access token and the secure session with the joining node.

4. The joining node and the KDC maintain the secure association, to support possible future communications. These especially include key management operations, such as retrieval of updated keying material or participation to a group rekeying process.
5. The joining node can communicate securely with the other group members, using the keying material provided in step 3.



(*) Defined in the ACE framework

Figure 2: Message Flow Upon New Node's Joining

3. Authorization to Join a Group

This section describes in detail the format of messages exchanged by the participants when a node requests access to a given group. This exchange is based on ACE [I-D.ietf-ace-oauth-authz].

As defined in [I-D.ietf-ace-oauth-authz], the Client requests the AS for the authorization to join the group through the KDC (see Section 3.1). If the request is approved and authorization is granted, the AS provides the Client with a proof-of-possession access token and parameters to securely communicate with the KDC (see Section 3.2).

Communications between the Client and the AS MUST be secured, according to what is defined by the used transport profile of ACE. The Content-Format used in the message also depends on the used transport profile of ACE. For example, it can be application/ace+cbor for the first two messages and application/cwt for the third message, which are defined in the ACE framework.

The transport profile of ACE also defines a number of details such as the communication and security protocols used with the KDC (see Appendix C of [I-D.ietf-ace-oauth-authz]).

Figure 3 gives an overview of the exchange described above.

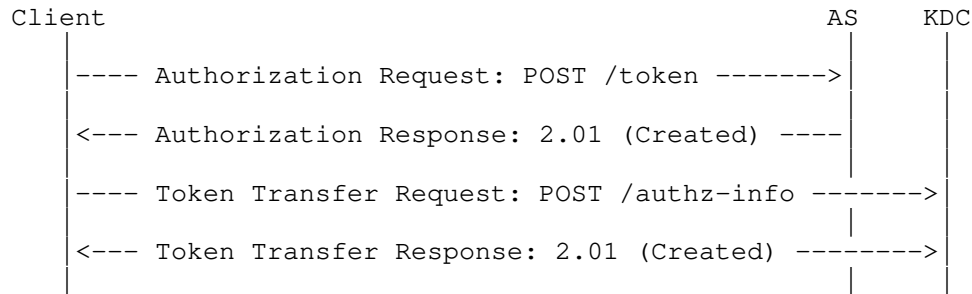


Figure 3: Message Flow of Join Authorization

3.1. Authorization Request

The Authorization Request sent from the Client to the AS is defined in Section 5.8.1 of [I-D.ietf-ace-oauth-authz] and MAY contain the following parameters, which, if included, MUST have format and value as specified below.

- * 'scope', specifying the name of the groups that the Client requests to access, and optionally the roles that the Client requests to have in those groups.

This parameter is encoded as a CBOR byte string, which wraps a CBOR array of one or more scope entries. All the scope entries are specified according to a same format, i.e. either the AIF format or the textual format defined below.

- If the AIF format is used, each scope entry is encoded as specified in [I-D.ietf-ace-aif]. The object identifier "Toid" corresponds to the group name and MUST be encoded as a CBOR text string. The permission set "Tperm" indicates the roles that the Client wishes to take in the group.

The AIF format is the default format for application profiles of this specification, and is preferable for those that aim to a compact encoding of scope. This is desirable especially for application profiles defining several roles, with the Client possibly requesting for multiple roles combined.

Figure 4 shows an example in CDDL notation [RFC8610] where scope uses the AIF format.

- If the textual format is used, each scope entry is a CBOR array formatted as follows.
 - o As first element, the group name, encoded as a CBOR text string.
 - o Optionally, as second element, the role or CBOR array of roles that the Client wishes to take in the group. This element is optional since roles may have been pre-assigned to the Client, as associated to its verifiable identity credentials. Alternatively, the application may have defined a single, well-known role for the target resource(s) and audience(s).

Figure 5 shows an example in CDDL notation where scope uses the textual format, with group name and role identifiers encoded as CBOR text strings.

It is REQUIRED of application profiles of this specification to specify the exact format and encoding of scope (REQ1). This includes defining the set of possible roles and their identifiers, as well as the corresponding encoding to use in the scope entries according to the used scope format.

If the application profile uses the AIF format, it is also REQUIRED to register its specific instance of "Toid" and "Tperm", as well as the corresponding Media Type and Content-Format, as per the guidelines in [I-D.ietf-ace-aif] (REQ2).

If the application profile uses the textual format, it MAY additionally specify CBOR values to use for abbreviating the role identifiers (OPT1).

* 'audience', with an identifier of the KDC.

As defined in [I-D.ietf-ace-oauth-authz], other additional parameters can be included if necessary.

```

gname = tstr

permissions = uint . bits roles

roles = &(amp;
    Requester: 1,
    Responder: 2,
    Monitor: 3,
    Verifier: 4
)

scope_entry = AIF_Generic<gname, permissions>

scope = << [ + scope_entry ] >>

```

Figure 4: Example of scope using the AIF format

```

gname = tstr

role = tstr

scope_entry = [ gname , ? ( role / [ 2*role ] ) ]

scope = << [ + scope_entry ] >>

```

Figure 5: Example of scope using the textual format, with the group name and role identifiers encoded as text strings

3.2. Authorization Response

The AS processes the Authorization Request as defined in Section 5.8.2 of [I-D.ietf-ace-oauth-authz], especially verifying that the Client is authorized to access the specified groups with the requested roles, or possibly a subset of those.

In case of successful verification, the Authorization Response sent from the AS to the Client is also defined in Section 5.8.2 of [I-D.ietf-ace-oauth-authz]. Note that the parameter 'expires_in' MAY be omitted if the application defines how the expiration time is communicated to the Client via other means, or if it establishes a default value.

Additionally, when included, the following parameter MUST have the corresponding values:

- * 'scope' has the same format and encoding of 'scope' in the Authorization Request, defined in Section 3.1. If this parameter is not present, the granted scope is equal to the one requested in Section 3.1.

The proof-of-possession access token (in 'access_token' above) MUST contain the following parameters:

- * a confirmation claim (see for example 'cnf' defined in Section 3.1 of [RFC8747] for CWT);
- * an expiration time claim (see for example 'exp' defined in Section 3.1.4 of [RFC8392] for CWT);
- * a scope claim (see for example 'scope' registered in Section 8.14 of [I-D.ietf-ace-oauth-authz] for CWT).

This claim specifies the same access control information as in the 'scope' parameter of the Authorization Response, if the parameter is present in the message, or as in the 'scope' parameter of the Authorization Request otherwise.

By default, this claim has the same encoding as the 'scope' parameter in the Authorization Request, defined in Section 3.1.

Optionally, an alternative extended format of scope defined in Section 7 can be used. This format explicitly signals the semantics used to express the actual access control information, and according to which this has to be parsed. This enables a Resource Server to correctly process a received access token, also in case:

- The Resource Server implements a KDC that supports multiple application profiles of this specification, using different scope semantics; and/or
- The Resource Server implements further services beyond a KDC for group communication, using different scope semantics.

If the Authorization Server is aware that this applies to the Resource Server for which the access token is issued, the Authorization Server SHOULD use the extended format of scope defined in Section 7.

The access token MAY additionally contain other claims that the transport profile of ACE requires, or other optional parameters.

When receiving an Authorization Request from a Client that was previously authorized, and for which the AS still owns a valid non-expired access token, the AS MAY reply with that token. Note that it is up to application profiles of ACE to make sure that re-posting the same token does not cause re-use of keying material between nodes (for example, that is done with the use of random nonces in [I-D.ietf-ace-oscore-profile]).

3.3. Token Transferring

The Client sends a Token Transfer Request to the KDC, i.e., a CoAP POST request including the access token and targeting the authz-info endpoint (see Section 5.10.1 of [I-D.ietf-ace-oauth-authz]).

Note that this request deviates from the one defined in [I-D.ietf-ace-oauth-authz], since it allows to ask the KDC for additional information concerning the public keys used in the group to ensure source authentication, as well as for possible additional group parameters.

The joining node MAY ask for this information from the KDC through the same Token Transfer Request. In this case, the message MUST have Content-Format set to application/ace+cbor defined in Section 8.16 of [I-D.ietf-ace-oauth-authz], and the message payload MUST be formatted as a CBOR map, which MUST include the access token. The CBOR map MAY additionally include the following parameter, which, if included, MUST have format and value as specified below.

- * 'sign_info' defined in Section 3.3.1, specifying the CBOR simple value 'null' (0xf6) to request information about the signature algorithm, signature algorithm parameters, signature key parameters and about the exact encoding of public keys used in the groups that the Client has been authorized to join.

Alternatively, such information may be pre-configured on the joining node, or may be retrieved by alternative means. For example, the joining node may have performed an early group discovery process and obtained the link to the associated group-membership resource at the KDC, together with attributes descriptive of the group configuration (see, e.g., [I-D.tiloca-core-oscore-discovery]).

After successful verification, the Client is authorized to receive the group keying material from the KDC and join the group. Hence, the KDC replies to the Client with a Token Transfer Response, i.e., a CoAP 2.01 (Created) response.

The Token Transfer Response MUST have Content-Format "application/ace+cbor", and its payload is a CBOR map. Note that this deviates from what is defined in the ACE framework, where the response from the authz-info endpoint is defined as conveying no payload (see Section 5.10.1 of [I-D.ietf-ace-oauth-authz]).

If the access token contains a role that requires the Client to send its own public key to the KDC when joining the group, the CBOR map MUST include the parameter 'kdcchallenge' defined in Section 3.3.2, specifying a dedicated challenge N_S generated by the KDC.

Later, when joining the group (see Section 4.3.1.1), the Client uses the 'kdcchallenge' value and additional information to build a proof-of-possession (PoP) input. This is in turn used to compute a PoP evidence, which the Client also provides to the Group Manager in order to prove possession of its own private key (see the 'client_cred_verify' parameter in Section 4.3.1).

The KDC MUST store the 'kdcchallenge' value associated to the Client at least until it receives a Joining Request from it (see Section 4.3.1.1), to be able to verify the PoP evidence provided during the join process, and thus that the Client possesses its own private key.

The same 'kdcchallenge' value MAY be reused several times by the Client, to generate a new PoP evidence, e.g., in case the Client provides the Group Manager with a new public key while being a group member (see Section 4.9.1.1), or joins a different group where it intends to use a different public key. Therefore, it is RECOMMENDED that the KDC keeps storing the 'kdcchallenge' value after the first join is processed as well. If the KDC has already discarded the 'kdcchallenge' value, that will trigger an error response with a newly generated 'kdcchallenge' value that the Client can use to restart the join process, as specified in Section 4.3.1.1.

If 'sign_info' is included in the Token Transfer Request, the KDC SHOULD include the 'sign_info' parameter in the Token Transfer Response, as per the format defined in Section 3.3.1. Note that the field 'id' of each 'sign_info_entry' specifies the name, or array of group names, for which that 'sign_info_entry' applies to. As an exception, the KDC MAY omit the 'sign_info' parameter in the Token Transfer Response even if 'sign_info' is included in the Token Transfer Request, in case none of the groups that the Client is authorized to join uses signatures to achieve source authentication.

Note that the CBOR map specified as payload of the 2.01 (Created) response may include further parameters, e.g., according to the used transport profile of ACE. Application profiles of this specification MAY define additional parameters to use within this exchange (OPT2).

Application profiles of this specification MAY define alternative specific negotiations of parameter values for the signature algorithm and signature keys, if 'sign_info' is not used (OPT3).

If allowed by the used transport profile of ACE, the Client may provide the Access Token to the KDC by other means than the Token Transfer Request. An example is the DTLS transport profile of ACE, where the Client can provide the access token to the KDC during the secure session establishment (see Section 3.3.2 of [I-D.ietf-ace-dtls-authorize]).

3.3.1. 'sign_info' Parameter

The 'sign_info' parameter is an OPTIONAL parameter of the request and response messages exchanged between the Client and the authz-info endpoint at the RS (see Section 5.10.1. of [I-D.ietf-ace-oauth-authz]).

This parameter allows the Client and the RS to exchange information about a signature algorithm and about public keys to accordingly use for signature verification. Its exact semantics and content are application specific.

In this specification and in application profiles building on it, this parameter is used to exchange information about the signature algorithm and about public keys to be used with it, in the groups indicated by the transferred access token as per its 'scope' claim (see Section 3.2).

When used in the Token Transfer Request sent to the KDC (see Section 3.3), the 'sign_info' parameter specifies the CBOR simple value 'null' (0xf6). This is done to ask for information about the signature algorithm and about the public keys used in the groups that the Client has been authorized to join - or to have a more restricted interaction as per its granted roles (e.g., the Client is an external signature verifier).

When used in the following Token Transfer Response from the KDC (see Section 3.3), the 'sign_info' parameter is a CBOR array of one or more elements. The number of elements is at most the number of groups that the Client has been authorized to join - or to have a more restricted interaction (see above). Each element contains information about signing parameters and about public keys for one or more groups, and is formatted as follows.

- * The first element 'id' is a group name or an array of group names, associated to groups for which the next four elements apply. In the following, each specified group name is referred to as 'gname'.
- * The second element 'sign_alg' is an integer or a text string if the POST request included the 'sign_info' parameter with value the CBOR simple value 'null' (0xf6), and indicates the signature algorithm used in the groups identified by the 'gname' values. It is REQUIRED of the application profiles to define specific values that this parameter can take (REQ3), selected from the set of signing algorithms of the COSE Algorithms registry [COSE.Algorithms].
- * The third element 'sign_parameters' is a CBOR array indicating the parameters of the signature algorithm used in the groups identified by the 'gname' values. Its content depends on the value of 'sign_alg'. It is REQUIRED of the application profiles to define the possible values and structure for the elements of this parameter (REQ4).
- * The fourth element 'sign_key_parameters' is a CBOR array indicating the parameters of the key used with the signature algorithm, in the groups identified by the 'gname' values. Its content depends on the value of 'sign_alg'. It is REQUIRED of the application profiles to define the possible values and structure for the elements of this parameter (REQ5).
- * The fifth element 'pub_key_enc' parameter is either a CBOR integer indicating the encoding of public keys used in the groups identified by the 'gname' values, or has value the CBOR simple value 'null' (0xf6) indicating that the KDC does not act as repository of public keys for group members. Its acceptable integer values are taken from the 'Label' column of the "COSE Header Parameters" registry [COSE.Header.Parameters]. It is REQUIRED of the application profiles to define specific values to use for this parameter, consistently with the acceptable formats of public keys (REQ6).

The CDDL notation [RFC8610] of the 'sign_info' parameter is given below.

```
sign_info = sign_info_req / sign_info_resp

sign_info_req = nil                                ; in the Token Transfer
                                                    ; Request to the KDC

sign_info_resp = [ + sign_info_entry ] ; in the Token Transfer
                                                    ; Response from the KDC

sign_info_entry =
[
  id : gname / [ + gname ],
  sign_alg : int / tstr,
  sign_parameters : [ any ],
  sign_key_parameters : [ any ],
  pub_key_enc = int / nil
]

gname = tstr
```

This format is consistent with every signature algorithm currently defined in [I-D.ietf-cose-rfc8152bis-algs], i.e., with algorithms that have only the COSE key type as their COSE capability. Appendix B describes how the format of each 'sign_info_entry' can be generalized for possible future registered algorithms having a different set of COSE capabilities.

3.3.2. 'kdcchallenge' Parameter

The 'kdcchallenge' parameter is an OPTIONAL parameter of response message returned from the authz-info endpoint at the RS, as defined in Section 5.10.1 of [I-D.ietf-ace-oauth-authz]. This parameter contains a challenge generated by the RS and provided to the Client.

In this specification and in application profiles building on it, the Client may use this challenge to prove possession of its own private key in the Joining Request (see the 'client_cred_verify' parameter in Section 4.3.1).

4. KDC Functionalities

This section describes the functionalities provided by the KDC, as related to the provisioning of the keying material as well as to the group membership management.

In particular, this section defines the interface available at the KDC; specifies the handlers of each resource provided by the KDC interface; and describes how Clients interact with those resources to join a group and to perform additional operations as group members.

As most important operation after transferring the access token to the KDC, the Client can perform a "Joining" exchange with the KDC, by specifying the group it requests to join (see Section 4.3.1.1). Then, the KDC verifies the access token and that the Client is authorized to join the specified group. If so, the KDC provides the Client with the keying material to securely communicate with the other members of the group.

Later on as a group member, the Client can also rely on the interface at the KDC to perform additional operations, consistently with the roles it has in the group.

4.1. Interface at the KDC

The KDC provides its interface by hosting the following resources. Note that the root url-path "ace-group" used hereafter is a default name; implementations are not required to use this name, and can define their own instead. The Interface Description (if=) Link Target Attribute value "ace.group" is registered in Section 11.5 and can be used to describe this interface.

If request messages sent to the KDC as well as success response messages from the KDC include a payload and specify a Content-Format, those messages MUST have Content-Format set to application/ace-groupcomm+cbor, defined in Section 11.2. CBOR labels for the message parameters are defined in Section 8.

- * /ace-group : this resource is invariant once established, and indicates that this specification is used. If other applications run on a KDC implementing this specification and use this same resource, those applications will collide, and a mechanism will be needed to differentiate the endpoints.

A Client can access this resource in order to retrieve a set of group names, each corresponding to one of the specified group identifiers. This operation is described in Section 4.2.1.1.

- * /ace-group/GROUPNAME : one such sub-resource to /ace-group is hosted for each group with name GROUPNAME that the KDC manages, and contains the symmetric group keying material for that group.

A Client can access this resource in order to join the group with name GROUPNAME, or later as a group member to retrieve the current group keying material. These operations are described in Section 4.3.1.1 and Section 4.3.2.1, respectively.

If the value of the GROUPNAME URI path and the group name in the access token scope ('gname' in Section 3.2) are not required to coincide, the KDC MUST implement a mechanism to map the GROUPNAME value in the URI to the group name, in order to refer to the correct group (REQ7).

- * /ace-group/GROUPNAME/pub-key : this resource is invariant once established, and contains the public keys of all the members of the group with name GROUPNAME.

This resource is created only in case the KDC acts as repository of public keys for group members.

A Client can access this resource in order to retrieve the public keys of other group members, in addition to when joining the group. That is, the Client can retrieve the public keys of all the current group members, or a subset of them by specifying filter criteria. These operations are described in Section 4.4.2.1 and Section 4.4.1.1, respectively.

Clients may be authorized to access this resource even without being group members, e.g., if authorized to be external signature verifiers for the group.

- * ace-group/GROUPNAME/kdc-pub-key : this resource is invariant once established, and contains the public key of the KDC for the group with name GROUPNAME.

This resource is created only in case the KDC has an associated public key and this is required for the correct group operation. It is REQUIRED of application profiles to define whether the KDC has such an associated public key (REQ8).

A Client can interact with this resource in order to retrieve the current public key of the KDC, in addition to when joining the group.

Clients may be authorized to access this resource even without being group members, e.g., if authorized to be external signature verifiers for the group.

- * /ace-group/GROUPNAME/policies : this resource is invariant once established, and contains the group policies of the group with name GROUPNAME.

A Client can access this resource as a group member in order to retrieve the group policies. This operation is described in Section 4.6.1.1.

- * /ace-group/GROUPNAME/num : this resource is invariant once established, and contains the current version number for the symmetric group keying material of the group with name GROUPNAME.

A Client can access this resource as a group member in order to retrieve the version number of the keying material currently used in the group. This operation is described in Section 4.7.1.1.

- * /ace-group/GROUPNAME/nodes/NODENAME : one such sub-resource of /ace-group/GROUPNAME is hosted for each group member of the group with name GROUPNAME. Each of such resources is identified by the node name NODENAME of the associated group member, and contains the group keying material and the individual keying material for that group member.

A Client as a group member can access this resource in order to retrieve the current group keying material together with its the individual keying material; request new individual keying material to use in the group; and leave the group. These operations are described in Section 4.8.1.1, Section 4.8.2.1, and Section 4.8.3.1, respectively.

- * /ace-group/GROUPNAME/nodes/NODENAME/pub-key : this resource is invariant once established, and contains the individual public keying material for the node with name NODENAME, as group member of the group with name GROUPNAME.

A Client can access this resource in order to upload at the KDC a new public key to use in the group. This operation is described in Section 4.9.1.1.

This resource is not created if the group member does not have individual public keying material to use in the group, or if the KDC does not store the public keys of group members.

The KDC is expected to fully provide the interface defined above. It is otherwise REQUIRED of the application profiles of this specification to indicate which resources are not hosted, i.e., which parts of the interface defined in this section are not supported by the KDC (REQ9). Application profiles of this specification MAY extend the KDC interface, by defining additional resources and their handlers.

It is REQUIRED of the application profiles of this specification to register a Resource Type for the root url-path (REQ10). This Resource Type can be used to discover the correct url to access at the KDC. This Resource Type can also be used at the GROUPNAME sub-resource, to indicate different application profiles for different groups.

It is REQUIRED of the application profiles of this specification to define what specific actions (e.g., CoAP methods) are allowed on each resource provided by the KDC interface, depending on whether the Client is a current group member; the roles that a Client is authorized to take as per the obtained access token (see Section 3.1); and the roles that the Client has as current group member (REQ11).

4.1.1. Operations Supported by Clients

It is expected that a Client minimally supports the following set of primary operations and corresponding interactions with the KDC.

- * FETCH request to ace-group/ , in order to retrieve group names associated to group identifiers.
- * POST and GET requests to ace-group/GROUPNAME/ , in order to join a group (POST) and later retrieve the current group key material as a group member (GET).
- * GET and FETCH requests to ace-group/GROUPNAME/pub-key , in order to retrieve the public keys of all the other group members (GET) or only some of them by filtering (FETCH). While retrieving public keys remains possible by using GET requests, retrieval by filtering allows to greatly limit the size of exchanged messages.
- * GET request to ace-group/GROUPNAME/num , in order to retrieve the current version of the group key material as a group member.
- * DELETE request to ace-group/GROUPNAME/nodes/NODENAME , in order to leave the group.

In addition, some Clients may rather not support the following set of secondary operations and corresponding interactions with the KDC. This can be specified, for instance, in compliance documents defining minimalistic Clients and their capabilities in specific deployments. In turn, these might also have to consider the used application profile of this specification.

- * GET request to `ace-group/GROUPNAME/kdc-pub-key` , in order to retrieve the current public key of the KDC, in addition to when joining the group. This is relevant only if the KDC has an associated public key and this is required for the correct group operation.
- * GET request to `ace-group/GROUPNAME/policies` , in order to retrieve the current group policies as a group member, in addition to when joining the group.
- * GET request to `ace-group/GROUPNAME/nodes/NODENAME`, in order to retrieve the current group keying material and individual keying material. The former can also be retrieved through a GET request to `ace-group/GROUPNAME/` (see above). The latter would not be possible to re-obtain as a group member.
- * PUT request to `ace-group/GROUPNAME/nodes/NODENAME` , in order to ask for new individual keying material. The Client would have to alternatively re-join the group through a POST request to `ace-group/GROUPNAME/` (see above). Furthermore, depending on its roles in the group or on the application profile of this specification, the Client might simply not be associated to any individual keying material.
- * POST request to `ace-group/GROUPNAME/nodes/NODENAME/pub-key` , in order to provide the KDC with a new public key. The Client would have to alternatively re-join the group through a POST request to `ace-group/GROUPNAME/` (see above). Furthermore, depending on its roles in the group, the Client might simply not have an associated public key to provide.

It is REQUIRED of application profiles of this specification to categorize possible newly defined operations for Clients into primary operations and secondary operations, and to provide accompanying considerations (REQ12).

4.1.2. Error Handling

Upon receiving a request from a Client, the KDC MUST check that it is storing a valid access token from that Client. If this is not the case, the KDC MUST reply with a 4.01 (Unauthorized) error response.

Unless the request targets the /ace-group resource, the KDC MUST check that it is storing a valid access token from that Client such that:

- * The scope specified in the access token includes a scope entry related to the group name GROUPNAME associated to targeted resource; and
- * The set of roles specified in that scope entry allows the Client to perform the requested operation on the targeted resource (REQ11).

In case the KDC stores a valid access token but the verifications above fail, the KDC MUST reply with a 4.03 (Forbidden) error response. This response MAY be an AS Request Creation Hints, as defined in Section 5.3 of [I-D.ietf-ace-oauth-authz], in which case the Content-Format MUST be set to application/ace+cbor.

If the request is not formatted correctly (e.g., required fields are not present or are not encoded as expected), the handler MUST reply with a 4.00 (Bad Request) error response.

If the request includes unknown or non-expected fields, the handler MUST silently ignore them and continue processing the request. Application profiles of this specification MAY define optional or mandatory payload formats for specific error cases (OPT4).

Some error responses from the KDC can have Content-Format set to application/ace-groupcomm+cbor. In such a case, the payload of the response MUST be a CBOR map, which includes the following fields.

- * 'error', with value a CBOR integer specifying the error occurred at the KDC. The value is taken from the "Value" column of the "ACE Groupcomm Errors" registry defined in Section 11.13 of this specification. This field MUST be present.
- * 'error_description', with value a CBOR text string specifying a human-readable diagnostic description of the error occurred at the KDC, written in English. The diagnostic text is intended for software engineers as well as for device and network operators, in order to aid debugging and provide context for possible intervention. The diagnostic message SHOULD be logged by the KDC. This field MAY be present, and it is unlikely relevant in an unattended setup where human intervention is not expected.

The 'error' and 'error_description' fields are defined as OPTIONAL to support for Clients (see Section 8). A Client supporting the 'error' parameter and able to understand the specified error may use that information to determine what actions to take next.

Section 9 of this specification defines an initial set of error identifiers, as possible values for the 'error' field. Application profiles of this specification inherit this initial set of error identifiers and MAY define additional value (OPT5).

4.2. /ace-group

This resource implements the FETCH handler.

4.2.1. FETCH Handler

The FETCH handler receives group identifiers and returns the corresponding group names and GROUPNAME URIs.

The handler expects a request with payload formatted as a CBOR map, which MUST contain the following fields:

- * 'gid', whose value is encoded as a CBOR array, containing one or more group identifiers. The exact encoding of group identifier MUST be specified by the application profile (REQ13). The Client indicates that it wishes to receive the group names and GROUPNAMEs of all groups having these identifiers.

The handler identifies the groups that are secured by the keying material identified by those group identifiers.

If all verifications succeed, the handler replies with a 2.05 (Content) response, whose payload is formatted as a CBOR map that MUST contain the following fields:

- * 'gid', whose value is encoded as a CBOR array, containing zero or more group identifiers. The handler indicates that those are the identifiers it is sending group names and GROUPNAMEs for. This CBOR array is a subset of the 'gid' array in the FETCH request.
- * 'gname', whose value is encoded as a CBOR array, containing zero or more group names. The elements of this array are encoded as text strings. Each element of index *i* of this CBOR array corresponds to the element of group identifier *i* in the 'gid' array.

- * 'guri', whose value is encoded as a CBOR array, containing zero or more URIs, each indicating a GROUPNAME resource. The elements of this array are encoded as text strings. Each element of index *i* of this CBOR array corresponds to the element of group identifier *i* in the 'gid' array.

If the KDC does not find any group associated to the specified group identifiers, the handler returns a response with payload formatted as a CBOR byte string of zero length.

Note that the KDC only verifies that the node is authorized by the AS to access this resource. Nodes that are not members of the group but are authorized to do signature verification on the group messages may be allowed to access this resource, if the application needs it.

4.2.1.1. Retrieve Group Names

In case the joining node only knows the group identifier of the group it wishes to join or about which it wishes to get update information from the KDC, the node can contact the KDC to request the corresponding group name and joining resource URI. The node can request several group identifiers at once. It does so by sending a CoAP FETCH request to the /ace-group endpoint at the KDC formatted as defined in Section 4.2.1.

Figure 6 gives an overview of the exchanges described above, and Figure 7 shows an example.

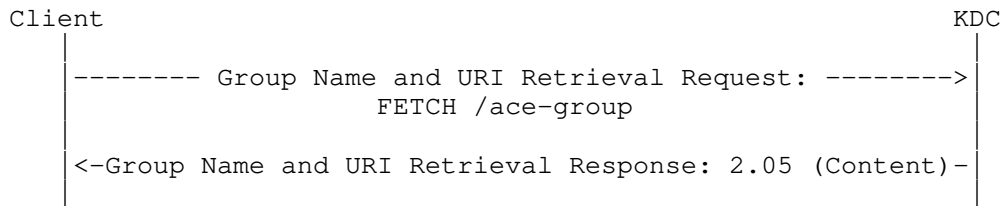


Figure 6: Message Flow of Group Name and URI Retrieval Request-Response

Request:

```
Header: FETCH (Code=0.05)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
  { "gid": [01, 02] }
```

Response:

```
Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
  { "gid": [01, 02], "gname": ["group1", "group2"],
    "guri": ["ace-group/g1", "ace-group/g2"] }
```

Figure 7: Example of Group Name and URI Retrieval Request-Response

4.3. /ace-group/GROUPNAME

This resource implements the POST and GET and handlers.

4.3.1. POST Handler

The POST handler processes the Joining Request sent by a Client to join a group, and returns a Joining Response as successful result of the joining process (see Section 4.3.1.1). At a high level, the POST handler adds the Client to the list of current group members, adds the public key of the Client to the list of the group members' public keys, and returns the symmetric group keying material for the group identified by GROUPNAME.

The handler expects a request with payload formatted as a CBOR map, which MAY contain the following fields, which, if included, MUST have format and value as specified below.

- * 'scope', with value the specific group that the Client is attempting to join, i.e., the group name, and the roles it wishes to have in the group. This value is a CBOR byte string wrapping one scope entry, as defined in Section 3.1.

- * `'get_pub_keys'`, if the Client wishes to receive the public keys of the current group members from the KDC. This parameter may be included in the Joining Request if the KDC stores the public keys of the group members, while it is not useful to include it if the Client obtains those public keys through alternative means, e.g., from the AS. Note that including this parameter might result in a following Joining Response of large size, which can be inconvenient for resource-constrained devices.

If the Client wishes to retrieve the public keys of all the current group members, the `'get_pub_keys'` parameter MUST encode the CBOR simple value `'null'` (0xf6). Otherwise, the `'get_pub_keys'` parameter MUST encode a non-empty CBOR array, containing the following three elements formatted as defined below.

- The first element, namely `'inclusion_flag'`, encodes the CBOR simple value True. That is, the Client indicates that it wishes to receive the public keys of all group members having their node identifier specified in the third element of the `'get_pub_keys'` array, namely `'id_filter'` (see below).
- The second element, namely `'role_filter'`, is a non-empty CBOR array. Each element of the array contains one role or a combination of roles for the group identified by GROUPNAME. That is, when the Joining Request includes a non-Null `'get_pub_keys'` parameter, the Client filters public keys based on node identifiers.

In particular, the Client indicates that it wishes to retrieve the public keys of all the group members having any of the single roles, or at least all of the roles indicated in any combination of roles. For example, the array `["role1", "role2+role3"]` indicates that the Client wishes to receive the public keys of all group members that have at least `"role1"` or at least both `"role2"` and `"role3"`.

- The third element, namely `'id_filter'`, is an empty CBOR array. That is, when the Joining Request includes a non-Null `'get_pub_keys'` parameter, the Client does not filter public keys based on node identifiers.

In fact, when first joining the group, the Client is not expected or capable to express a filter based on node identifiers of other group members. Instead, when already a group member and sending a Joining Request to re-join, the Client is not expected to include the 'get_pub_keys' parameter in the Joining Request altogether, since it can rather retrieve public keys associated to specific group identifiers as defined in Section 4.4.1.1.

The CDDL definition [RFC8610] of 'get_pub_keys' is given in Figure 8, using as example encoding: node identifier encoded as a CBOR byte string; role identifier encoded as a CBOR text string, and combination of roles encoded as a CBOR array of roles.

Note that, for this handler, 'inclusion_flag' is always set to true, the array of roles 'role_filter' is always non-empty, while the array of node identifiers 'id_filter' is always empty. However, this is not necessarily the case for other handlers using the 'get_pub_keys' parameter.

```
inclusion_flag = bool

role = tstr
comb_role = [ 2*role ]
role_filter = [ *(role / comb_role) ]

id = bstr
id_filter = [ *id ]

get_pub_keys = null / [ inclusion_flag, role_filter, id_filter]
```

Figure 8: CDDL definition of get_pub_keys, using as example node identifier encoded as bstr and role as tstr

- * 'client_cred', encoded as a CBOR byte string, with value the original binary representation of the Client's public key. This parameter is used if the KDC is managing (collecting from/ distributing to the Client) the public keys of the group members, and if the Client's role in the group will require for it to send messages to one or more group members. It is REQUIRED of the application profiles to define the specific formats that are acceptable to use for encoding public keys in the group (REQ6).
- * 'nonce', encoded as a CBOR byte string, and including a dedicated nonce N_C generated by the Client. This parameter MUST be present if the 'client_cred' parameter is present.

- * `'client_cred_verify'`, encoded as a CBOR byte string. This parameter MUST be present if the `'client_cred'` parameter is present and no public key associated to the Client's token can be retrieved for that group.

This parameter contains a proof-of-possession (PoP) evidence computed by the Client over the following PoP input: the scope (encoded as CBOR byte string), concatenated with `N_S` (encoded as CBOR byte string) concatenated with `N_C` (encoded as CBOR byte string), where:

- scope is the CBOR byte string either specified in the `'scope'` parameter above, if present, or as a default scope that the handler is expected to understand, if omitted.
- `N_S` is the challenge received from the KDC in the `'kdcchallenge'` parameter of the 2.01 (Created) response to the Token Transfer Request (see Section 3.3), encoded as a CBOR byte string.
- `N_C` is the nonce generated by the Client and specified in the `'cnonce'` parameter above, encoded as a CBOR byte string.

An example of PoP input to compute `'client_cred_verify'` using CBOR encoding is given in Figure 9.

A possible type of PoP evidence is a signature, that the Client computes by using its own private key, whose corresponding public key is specified in the `'client_cred'` parameter. Application profiles of this specification MUST specify the exact approaches used to compute the PoP evidence to include in `'client_cred_verify'`, and MUST specify which of those approaches is used in which case (REQ14).

If the token was not provided to the KDC through a Token Transfer Request (e.g., it is used directly to validate TLS instead), it is REQUIRED of the specific application profile to define how the challenge `N_S` is generated (REQ15).

- * `'pub_keys_repos'`, which can be present if the format of the Client's public key in the `'client_cred'` parameter is a certificate. In such a case, this parameter has as value the URI of the certificate. This parameter is encoded as a CBOR text string. Alternative specific encodings of this parameter MAY be defined in applications of this specification (OPT6).

- * 'control_uri', with value a full URI, encoded as a CBOR text string. A default url-path is /ace-group/GROUPNAME/node, although implementations can use different ones instead. The URI MUST NOT have url-path ace-group/GROUPNAME.

If 'control_uri' is specified in the Joining Request, the Client acts as a CoAP server and hosts a resource at this specific URI. The KDC MAY use this URI to send CoAP requests to the Client (acting as CoAP server in this exchange), for example for one-to-one provisioning of new group keying material when performing a group rekeying (see Section 4.8.1.1), or to inform the Client of its removal from the group Section 5.

In particular, this resource is intended for communications concerning exclusively the group whose group name GROUPNAME is specified in the 'scope' parameter. If the KDC does not implement mechanisms using this resource for that group, it can ignore this parameter. Other additional functionalities of this resource MAY be defined in application profiles of this specifications (OPT7).

scope, N_S, and N_C expressed in CBOR diagnostic notation:

```
scope = h'826667726F7570316673656E646572'
N_S   = h'018a278f7faab55a'
N_C   = h'25a8991cd700ac01'
```

scope, N_S, and N_C as CBOR encoded byte strings:

```
scope = 0x4f826667726F7570316673656E646572
N_S   = 0x48018a278f7faab55a
N_C   = 0x4825a8991cd700ac01
```

PoP input:

```
0x4f 826667726F7570316673656E646572
48 018a278f7faab55a 48 25a8991cd700ac01
```

Figure 9: Example of PoP input to compute 'client_cred_verify' using CBOR encoding

If the request does not include a 'scope' field, the KDC is expected to understand with what roles the Client is requesting to join the group. For example, as per the access token, the Client might have been granted access to the group with only one role. If the KDC cannot determine which exact scope should be considered for the Client, it MUST reply with a 4.00 (Bad Request) error response.

The handler considers the scope specified in the access token associated to the Client, and checks the scope entry related to the group with name GROUPNAME associated to the endpoint. In particular,

the handler checks whether the set of roles specified in that scope entry includes all the roles that the Client wishes to have in the group as per the Joining Request. If this is not the case, the KDC MUST reply with a 4.03 (Forbidden) error response.

If the KDC manages the group members' public keys, the handler checks if one is included in the 'client_cred' field. If so, the KDC retrieves the public key and performs the following actions.

- * If the access token was provided through a Token Transfer Request (see Section 3.3) but the KDC cannot retrieve the 'kdcchallenge' associated to this Client (see Section 3.3), the KDC MUST reply with a 4.00 Bad Request error response, which MUST also have Content-Format application/ace-groupcomm+cbor. The payload of the error response is a CBOR map including a newly generated 'kdcchallenge' value. This is specified in the 'kdcchallenge' parameter.
- * The KDC checks the public key to be valid for the group identified by GROUPNAME. That is, it checks that the public key is encoded according to the format used in the group, is intended for the public key algorithm used in the group, and is aligned with the possible associated parameters used in the group.

If this verification fails, the handler MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 2 ("Public key incompatible with the group configuration").

- * The KDC verifies the PoP evidence contained in the 'client_cred_verify' field. Application profiles of this specification MUST specify the exact approaches used to verify the PoP evidence, and MUST specify which of those approaches is used in which case (REQ14).

If the PoP evidence does not pass verification, the handler MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 3 ("Invalid Proof-of-Possession evidence").

If no public key is included in the 'client_cred' field, the handler checks if a public key is already associated to the received access token and to the group identified by GROUPNAME (see also Section 4.3.1.1). Note that the same joining node may use different public keys in different groups, and all those public keys would be associate to the same access token.

If an eligible public key for the Client is neither present in the 'client_cred' field nor retrieved from the stored ones at the KDC, it is RECOMMENDED that the handler stops the processing and replies with a 4.00 (Bad Request) error response. Applications profiles MAY define alternatives (OPT8).

If, regardless the reason, the KDC replies with a 4.00 (Bad Request) error response, this response MAY have Content-Format set to application/ace-groupcomm+cbor and have a CBOR map as payload. For instance, the CBOR map can include a 'sign_info' parameter formatted as 'sign_info_res' defined in Section 3.3.1, with the 'pub_key_enc' element set to the CBOR simple value 'null' (0xf6) if the Client sent its own public key and the KDC is not set to store public keys of the group members.

If all the verifications above succeed, the KDC proceeds as follows.

First, only in case the Client is not already a group member, the handler performs the following actions:

- * The handler adds the Client to the list of current members of the group.
- * The handler assigns a name NODENAME to the Client, and creates a sub-resource to /ace-group/GROUPNAME at the KDC, i.e., "/ace-group/GROUPNAME/nodes/NODENAME".
- * The handler associates the node identifier NODENAME to the access token and the secure session for the Client.

Then, the handler performs the following actions.

- * If the KDC manages the group members' public keys:
 - The handler associates the retrieved Client's public key to the tuple composed of the node name NODENAME, the group name GROUPNAME and the received access token.
 - The handler adds the retrieved Client's public key to the stored list of public keys stored for the group identified by GROUPNAME. If such list already includes a public key for the Client, but a different public key is specified in the 'client_cred' field, then the handler MUST replace the old public key in the list with the one specified in the 'client_cred' field.

- * If the application requires backward security or if the used application profile prescribes so, the KDC MUST generate new group keying material and securely distribute it to the current group members (see Section 6).
- * The handler returns a successful Joining Response as defined below, containing the symmetric group keying material; the group policies; and the public keys of the current members of the group, if the KDC manages those and the Client requested them.

The Joining Response MUST have response code 2.01 (Created) if the Client has been added to the list of group members in this joining exchange (see above), or 2.04 (Changed) otherwise, i.e., if the Client is re-joining the group without having left it.

The Joining Response message MUST include the Location-Path CoAP option, specifying the URI path to the sub-resource associated to the Client, i.e. `"/ace-group/GROUPNAME/nodes/NODENAME"`.

The Joining Response message MUST have Content-Format `application/ace-groupcomm+cbor`. The payload of the response is formatted as a CBOR map, which MUST contain the following fields and values.

- * `'gkty'`, identifying the key type of the `'key'` parameter. The set of values can be found in the "Key Type" column of the "ACE Groupcomm Key Types" registry. Implementations MUST verify that the key type matches the application profile being used, if present, as registered in the "ACE Groupcomm Key Types" registry.
- * `'key'`, containing the keying material for the group communication, or information required to derive it.
- * `'num'`, containing the version number of the keying material for the group communication, formatted as an integer. This is a strictly monotonic increasing field. The application profile MUST define the initial version number (REQ16).

The exact format of the `'key'` value MUST be defined in applications of this specification (REQ17), as well as values of `'gkty'` accepted by the application (REQ18). Additionally, documents specifying the key format MUST register it in the "ACE Groupcomm Key Types" registry defined in Section 11.8, including its name, type and application profile to be used with.

Name	Key Type Value	Profile	Description
Reserved	0		This value is reserved

Figure 10: Key Type Values

The response SHOULD contain the following parameter:

- * `'exp'`, with value the expiration time of the keying material for the group communication, encoded as a CBOR unsigned integer. This field contains a numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds, analogous to what specified for `NumericDate` in Section 2 of [RFC7519]. Group members MUST stop using the keying material to protect outgoing messages and retrieve new keying material at the time indicated in this field.

Optionally, the response MAY contain the following parameters, which, if included, MUST have format and value as specified below.

- * `'ace-groupcomm-profile'`, with value a CBOR integer that MUST be used to uniquely identify the application profile for group communication. Applications of this specification MUST register an application profile identifier and the related value for this parameter in the "ACE Groupcomm Profiles" registry (REQ19).
- * `'pub_keys'`, MUST be present if `'get_pub_keys'` was present in the request, otherwise it MUST NOT be present. This parameter is a CBOR array specifying the public keys of the group members, i.e., of all of them or of the ones selected according to the `'get_pub_keys'` parameter in the request. In particular, each element of the array is a CBOR byte string, with value the original binary representation of a group member's public key. It is REQUIRED of the application profiles to define the specific formats of public keys that are acceptable to use in the group (REQ6).
- * `'peer_roles'`, MUST be present if `'pub_keys'` is also present, otherwise it MUST NOT be present. This parameter is a CBOR array of n elements, with n the number of public keys included in the `'pub_keys'` parameter (at most the number of members in the group). The i -th element of the array specifies the role (or CBOR array of roles) that the group member associated to the i -th public key in `'pub_keys'` has in the group. In particular, each array element is encoded as the role element of a scope entry, as defined in Section 3.1.

- * `'peer_identifiers'`, MUST be present if `'pub_keys'` is also present, otherwise it MUST NOT be present. This parameter is a CBOR array of `n` elements, with `n` the number of public keys included in the `'pub_keys'` parameter (at most the number of members in the group). The `i`-th element of the array specifies the node identifier that the group member associated to the `i`-th public key in `'pub_keys'` has in the group. In particular, the `i`-th array element is encoded as a CBOR byte string, with value the node identifier of the group member.
- * `'group_policies'`, with value a CBOR map, whose entries specify how the group handles specific management aspects. These include, for instance, approaches to achieve synchronization of sequence numbers among group members. The elements of this field are registered in the "ACE Groupcomm Policies" registry. This specification defines the three elements "Sequence Number Synchronization Methods", "Key Update Check Interval" and "Expiration Delta", which are summarized in Figure 11. Application profiles that build on this document MUST specify the exact content format and default value of included map entries (REQ20).

Name	CBOR label	CBOR type	Description	Reference
Sequence Number Synchronization Method	TBD	tstr/int	Method for recipient group members to synchronize with sequence numbers of of sender group members. Its value is taken from the 'Value' column of the Sequence Number Synchronization Method registry	[[this document]]
Key Update Check Interval	TBD	int	Polling interval in seconds, for group members to check at the KDC if the latest group keying material is the one that they own	[[this document]]
Expiration Delta	TBD	uint	Number of seconds from 'exp' until the specified UTC date/time after which group members MUST stop using the group keying material they own to verify incoming messages	[[this document]]

Figure 11: ACE Groupcomm Policies

- * 'kdc_cred', encoded as a CBOR byte string, with value the original binary representation of the KDC's public key. This parameter is used if the KDC has an associated public key and this is required for the correct group operation. It is REQUIRED of application profiles to define whether the KDC has a public key and if this has to be provided through the 'kdc_cred' parameter (REQ8).

In such a case, the KDC's public key MUST have the same format used for the public keys of the group members. It is REQUIRED of the application profiles to define the specific formats that are acceptable to use for encoding public keys in the group (REQ6).

- * 'kdc_nonce', encoded as a CBOR byte string, and including a dedicated nonce N_KDC generated by the KDC. This parameter MUST be present if the 'kdc_cred' parameter is present.
- * 'kdc_cred_verify' parameter, encoded as a CBOR byte string. This parameter MUST be present if the 'kdc_cred' parameter is present.

This parameter contains a proof-of-possession (PoP) evidence computed by the KDC over the nonce N_KDC, which is specified in the 'kdc_nonce' parameter and taken as PoP input.

A possible type of PoP evidence is a signature, that the KDC computes by using its own private key, whose corresponding public key is specified in the 'kdc_cred' parameter. Application profiles of this specification MUST specify the exact approaches used by the KDC to compute the PoP evidence to include in 'kdc_cred_verify', and MUST specify which of those approaches is used in which case (REQ21).

- * 'rekeying_scheme', identifying the rekeying scheme that the KDC uses to provide new group keying material to the group members. This parameter is encoded as a CBOR integer, whose value is taken from the "Value" column of the "ACE Groupcomm Rekeying Schemes" registry defined in Section 11.14 of this specification.

Value	Name	Description	Reference
0	Point-to-Point	The KDC individually targets each node to rekey, using the pairwise secure communication association with that node	[this document]

Figure 12: ACE Groupcomm Rekeying Schemes

Application profiles of this specification MAY define a default group rekeying scheme, to refer to in case the 'rekeying_scheme' parameter is not included in the Joining Response (OPT9).

- * 'mgt_key_material', encoded as a CBOR byte string and containing the specific administrative keying material that the joining node requires in order to participate in the group rekeying process

performed by the KDC. This parameter MUST NOT be present if the 'rekeying_scheme' parameter is not present and the application profile does not specify a default group rekeying scheme to use in the group. Some simple rekeying scheme may not require specific administrative keying material to be provided, e.g., the basic "Point-to-Point" group rekeying scheme (see Section 6.1).

In more advanced group rekeying schemes, the administrative keying material can be composed of multiple keys organized, for instance, into a logical tree hierarchy, whose root key is the only administrative key shared by all the group members. In such a case, each group member is exclusively associated to one leaf key in the hierarchy, and owns only the administrative keys from the associated leaf key all the way up along the path to the root key. That is, different group members can be provided with a different subset of the overall administrative keying material.

It is expected from separate documents to define how the advanced group rekeying scheme possibly indicated in the 'rekeying_scheme' parameter is used by an application profile of this specification. This includes defining the format of the administrative keying material to specify in 'mgt_key_material', consistently with the group rekeying scheme and the application profile in question.

- * 'control_group_uri', with value a full URI, encoded as a CBOR text string. The URI MUST specify addressing information intended to reach all the members in the group. For example, this can be a multicast IP address, optionally together with a port number (which defaults to 5683 if omitted). The URI MUST include GROUPNAME in the url-path. A default url-path is /ace-group/GROUPNAME, although implementations can use different ones instead. The URI MUST NOT have url-path ace-group/GROUPNAME/node.

If 'control_group_uri' is included in the Joining Response, the Clients supporting this parameter act as CoAP servers, host a resource at this specific URI, and listen to the specified addressing information.

The KDC MAY use this URI to send one-to-many CoAP requests to the Client group members (acting as CoAP servers in this exchange), for example for one-to-many provisioning of new group keying material when performing a group rekeying (see Section 4.8.1.1), or to inform the Clients of their removal from the group
Section 5.

In particular, this resource is intended for communications concerning exclusively the group whose group name GROUPNAME is specified in the 'scope' parameter. If the KDC does not implement

mechanisms using this resource for that group, it can ignore this parameter. Other additional functionalities of this resource MAY be defined in application profiles of this specifications (OPT10).

If the Joining Response includes the 'kdc_cred_verify' parameter, the Client verifies the conveyed PoP evidence and considers the group joining unsuccessful in case of failed verification. Application profiles of this specification MUST specify the exact approaches used by the Client to verify the PoP evidence in 'kdc_cred_verify', and MUST specify which of those approaches is used in which case (REQ21).

Specific application profiles that build on this document MUST specify the communication protocol that members of the group use to communicate with each other (REQ22) and how exactly the keying material is used to protect the group communication (REQ23).

4.3.1.1. Join the Group

Figure 13 gives an overview of the Joining exchange between Client and KDC, when the Client first joins a group, while Figure 14 shows an example.

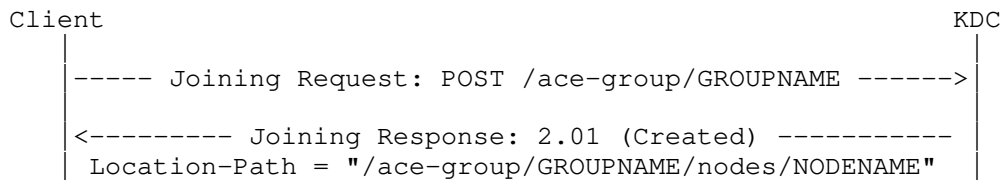


Figure 13: Message Flow of the Joining Exchange

Request:

```
Header: POST (Code=0.02)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation,
        with PUB_KEY and POP_EVIDENCE being CBOR byte strings):
{ "scope": << [ "group1", ["sender", "receiver"] ] >> ,
  "get_pub_keys": [true, ["sender"], []], "client_cred": PUB_KEY,
  "cnonce": h'6df49c495409a9b5', "client_cred_verify": POP_EVIDENCE }
```

Response:

```
Header: Created (Code=2.01)
Content-Format: "application/ace-groupcomm+cbor"
Location-Path: "kdc.example.com"
Location-Path: "g1"
Location-Path: "nodes"
Location-Path: "c101"
Payload (in CBOR diagnostic notation,
        with KEY being a CBOR byte strings):
{ "gkty": 13, "key": KEY, "num": 12, "exp": 1609459200,
  "pub_keys": [ PUB_KEY1, PUB_KEY2 ],
  "peer_roles": ["sender", ["sender", "receiver"]],
  "peer_identifiers": [ ID1, ID2 ] }
```

Figure 14: Example of First Exchange for Group Joining

If not previously established, the Client and the KDC MUST first establish a pairwise secure communication channel (REQ24). This can be achieved, for instance, by using a transport profile of ACE. The Joining exchange MUST occur over that secure channel. The Client and the KDC MAY use that same secure channel to protect further pairwise communications that must be secured.

The secure communication protocol is REQUIRED to establish the secure channel between Client and KDC by using the proof-of-possession key bound to the access token. As a result, the proof-of-possession to bind the access token to the Client is performed by using the proof-of-possession key bound to the access token for establishing secure communication between the Client and the KDC.

To join the group, the Client sends a CoAP POST request to the /ace-group/GROUPNAME endpoint at the KDC, where GROUPNAME is the group name of the group to join, formatted as specified in Section 4.3.1. This group name is the same as in the scope entry corresponding to

that group, specified in the 'scope' parameter of the Authorization Request/Response, or it can be retrieved from it. Note that, in case of successful joining, the Client will receive the URI to retrieve individual keying material and to leave the group in the Location-Path option of the response.

If the node is joining a group for the first time, and the KDC maintains the public keys of the group members, the Client is REQUIRED to send its own public key and proof-of-possession (PoP) evidence in the Joining Request (see the 'client_cred' and 'client_cred_verify' parameters in Section 4.3.1). The request is accepted only if both public key is provided and the PoP evidence is successfully verified.

If a node re-joins a group as authorized by the same access token and using the same public key, it can omit the public key and the PoP evidence, or just the PoP evidence, from the Joining Request. Then, the KDC will be able to retrieve the node's public key associated to the access token for that group. If the public key has been discarded, the KDC replies with 4.00 (Bad Request) error response, as specified in Section 4.3.1. If a node re-joins a group but wants to update its own public key, it needs to include both its public key and the PoP evidence in the Joining Request like when it joined the group for the first time.

4.3.2. GET Handler

The GET handler returns the symmetric group keying material for the group identified by GROUPNAME.

The handler expects a GET request.

In addition to what is defined in Section 4.1.2, the handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler replies with a 2.05 (Content) response containing the symmetric group keying material. The payload of the response is formatted as a CBOR map which MUST contain the parameters 'gkty', 'key' and 'num' specified in Section 4.3.1.

Each of the following parameters specified in Section 4.3.1 MUST also be included in the payload of the response, if they are included in the payload of the Joining Responses sent for the group: 'rekeying_scheme', 'mgt_key_material'.

The payload MAY also include the parameters 'ace-groupcomm-profile' and 'exp' parameters specified in Section 4.3.1.

4.3.2.1. Retrieve Group Keying Material

A node in the group can contact the KDC to retrieve the current group keying material, by sending a CoAP GET request to the /ace-group/GROUPNAME endpoint at the KDC, where GROUPNAME is the group name.

Figure 15 gives an overview of the Joining exchange between Client and KDC, when the Client first joins a group, while Figure 16 shows an example.

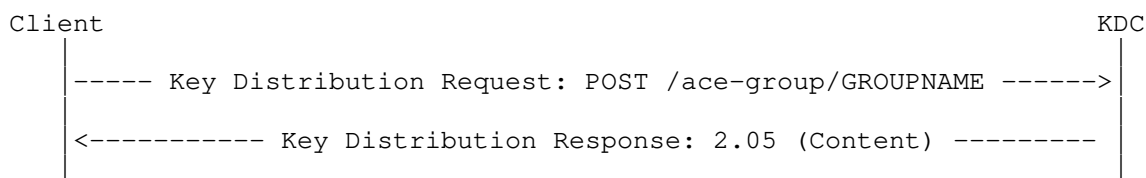


Figure 15: Message Flow of Key Distribution Request-Response

Request:

```

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Payload: -
  
```

Response:

```

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation,
        with KEY being a CBOR byte strings):
{ "gkty": 13, "key": KEY, "num": 12 }
  
```

Figure 16: Example of Key Distribution Request-Response

4.4. /ace-group/GROUPNAME/pub-key

This resource implements the GET and FETCH handlers.

4.4.1. FETCH Handler

The FETCH handler receives identifiers of group members for the group identified by GROUPNAME and returns the public keys of such group members.

The handler expects a request with payload formatted as a CBOR map, that MUST contain the following field.

* 'get_pub_keys', whose value is encoded as in Section 4.3.1 with the following modifications.

- The arrays 'role_filter' and 'id_filter' MUST NOT both be empty, i.e., in CBOR diagnostic notation: [bool, [], []]. If the 'get_pub_keys' parameter has such a format, the request MUST be considered malformed, and the KDC MUST reply with a 4.00 (Bad Request) error response.

Note that a group member can retrieve the public keys of all the current group members by sending a GET request to the same KDC resource instead (see Section 4.4.2.1).

- The element 'inclusion_flag' encodes the CBOR simple value True if the third element 'id_filter' specifies an empty CBOR array, or if the Client wishes to receive the public keys of the nodes having their node identifier specified in 'id_filter' (i.e., selection by inclusive filtering). Instead, this element encodes the CBOR simple value False if the Client wishes to receive the public keys of the nodes not having the node identifiers specified in the third element 'id_filter' (i.e., selection by exclusive filtering).
- The array 'role_filter' can be empty, if the Client does not wish to filter the requested public keys based on the roles of the group members.
- The array 'id_filter' contains zero or more node identifiers of group members, for the group identified by GROUPNAME. The Client indicates that it wishes to receive the public keys of the nodes having or not having these node identifiers, in case the 'inclusion_flag' element encodes the CBOR simple value True or False, respectively. The array 'id_filter' may be empty, if the Client does not wish to filter the requested public keys based on the node identifiers of the group members.

Note that, in case the 'role_filter' array and the 'id_filter' array are both non-empty:

- * If the 'inclusion_flag' encodes the CBOR simple value True, the handler returns the public keys of group members whose roles match with 'role_filter' and/or having their node identifier specified in 'id_filter'.
- * If the 'inclusion_flag' encodes the CBOR simple value False, the handler returns the public keys of group members whose roles match with 'role_filter' and, at the same time, not having their node identifier specified in 'id_filter'.

The specific format of public keys as well as identifiers, roles and combination of roles of group members MUST be specified by It is REQUIRED of application profiles of this specification (REQ1, REQ6, REQ25).

The handler identifies the public keys of the current group members for which either:

- * the role identifier matches with one of those indicated in the request; note that the request can contain a "combination of roles", where the handler select all group members who have all roles included in the combination.
- * the node identifier matches with one of those indicated in the request.

If all verifications succeed, the handler returns a 2.05 (Content) message response with payload formatted as a CBOR map, containing only the following parameters from Section 4.3.1.

- * 'num', which encodes the version number of the current group keying material.
- * 'pub_keys', which encodes the list of public keys of the selected group members.
- * 'peer_roles', which encodes the role (or CBOR array of roles) that each of the selected group members has in the group.
- * 'peer_identifiers', which encodes the node identifier that each of the selected group members has in the group.

The specific format of public keys as well as of node identifiers of group members is specified by the application profile (REQ6, REQ25).

If the KDC does not store any public key associated to the specified node identifiers, the handler returns a response with payload formatted as a CBOR byte string of zero length.

The handler MAY enforce one of the following policies, in order to handle possible node identifiers that are included in the 'id_filter' element of the 'get_pub_keys' parameter of the request but are not associated to any current group member. Such a policy MUST be specified by the application profile (REQ26).

- * The KDC silently ignores those node identifiers.
- * The KDC retains public keys of group members for a given amount of time after their leaving, before discarding them. As long as such public keys are retained, the KDC provides them to a requesting Client.

If the KDC adopts this policy, the application profile MUST also specify the amount of time during which the KDC retains the public key of a former group member after its leaving, possibly on a per-member basis.

Note that this resource handler only verifies that the node is authorized by the AS to access this resource. Nodes that are not members of the group but are authorized to do signature verifications on the group messages may be allowed to access this resource, if the application needs it.

4.4.1.1. Retrieve a Subset of Public Keys in the Group

In case the KDC maintains the public keys of group members, a node in the group can contact the KDC to request the public keys, roles and node identifiers of a specified subset of group members, by sending a CoAP FETCH request to the /ace-group/GROUPNAME/pub-key endpoint at the KDC, where GROUPNAME is the group name, and formatted as defined in Section 4.4.1.

Figure 17 gives an overview of the exchange mentioned above, while Figure 18 shows an example of such an exchange.

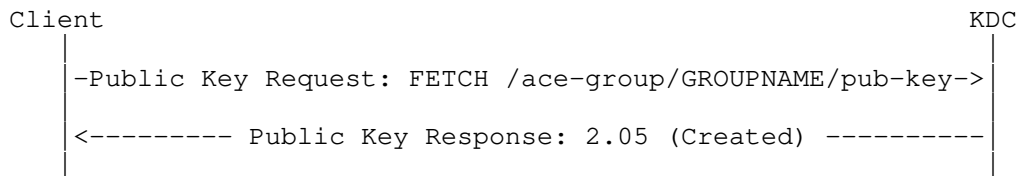


Figure 17: Message Flow of Public Key Exchange to Request the Public Keys of Specific Group Members

Request:

```
Header: FETCH (Code=0.05)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "pub-key"
Content-Format: "application/ace-groupcomm+cbor"
Payload:
  { "get_pub_keys": [true, [], [ ID3 ]] }
```

Response:

```
Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
  { "pub_keys": [ PUB_KEY3 ],
    "peer_roles": [ "receiver" ],
    "peer_identifiers": [ ID3 ] }
```

Figure 18: Example of Public Key Exchange to Request the Public Keys of Specific Group Members

4.4.2. GET Handler

The handler expects a GET request.

If all verifications succeed, the KDC replies with a 2.05 (Content) response as in the FETCH handler in Section 4.4.1, but specifying in the payload the public keys of all the group members, together with their roles and node identifiers.

4.4.2.1. Retrieve All Public Keys in the Group

In case the KDC maintains the public keys of group members, a group or an external signature verifier can contact the KDC to request the public keys, roles and node identifiers of all the current group members, by sending a CoAP GET request to the /ace-group/GROUPNAME/pub-key endpoint at the KDC, where GROUPNAME is the group name.

Figure 19 gives an overview of the message exchange, while Figure 20 shows an example of such an exchange.

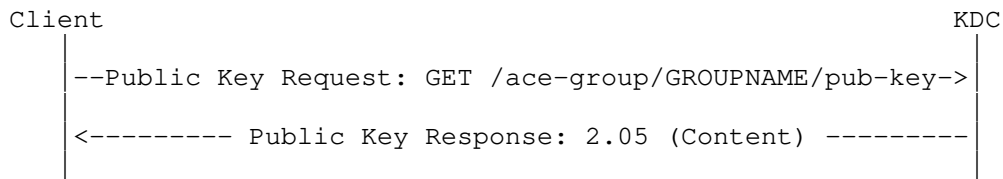


Figure 19: Message Flow of Public Key Exchange to Request the Public Keys of all the Group Members

Request:

```

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "pub-key"
Payload: -
    
```

Response:

```

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
{ "num": 5,
  "pub_keys": [ PUB_KEY1, PUB_KEY2, PUB_KEY3 ],
  "peer_roles": ["sender", ["sender", "receiver"], "receiver"],
  "peer_identifiers": [ ID1, ID2, ID3 ] }
    
```

Figure 20: Example of Public Key Exchange to Request the Public Keys of all the Group Members

4.5. ace-group/GROUPNAME/kdc-pub-key

This resource implements a GET handler.

4.5.1. GET Handler

The handler expects a GET request.

If all verifications succeed, the handler returns a 2.05 (Content) message containing the KDC's public key together with a proof-of-possession (PoP) evidence. The response MUST have Content-Format set to application/ace-groupcomm+cbor. The payload of the response is a CBOR map, which includes the following fields.

- * The 'kdc_cred' parameter, specifying the KDC's public key. This parameter is encoded like the 'kdc_cred' parameter in the Joining Response (see Section 4.3.1).
- * The 'kdc_nonce' parameter, specifying a nonce generated by the KDC. This parameter is encoded like the 'kdc_nonce' parameter in the Joining Response (see Section 4.3.1).
- * The 'kdc_cred_verify' parameter, specifying a PoP evidence computed by the KDC. This parameter is encoded like the 'kdc_cred_verify' parameter in the Joining Response (see Section 4.3.1).

The PoP evidence is computed over the nonce specified in the 'kdc_nonce' parameter and taken as PoP input, by means of the same method used when preparing the Joining Response (see Section 4.3.1). Application profiles of this specification MUST specify the exact approaches used by the KDC to compute the PoP evidence to include in 'kdc_cred_verify', and MUST specify which of those approaches is used in which case (REQ21).

4.5.1.1. Retrieve the KDC's Public Key

In case the KDC has an associated public key as required for the correct group operation, a group member or an external signature verifier can contact the KDC to request the KDC's public key, by sending a CoAP GET request to the /ace-group/GROUPNAME/kdc-pub-key endpoint at the KDC, where GROUPNAME is the group name.

Upon receiving the 2.05 (Content) response, the Client retrieves the KDC's public key from the 'kdc_cred' parameter, and MUST verify the proof-of-possession (PoP) evidence specified in the 'kdc_cred_verify' parameter. In case of successful verification of the PoP evidence, the Client MUST store the obtained KDC's public key and replace the currently stored one.

The PoP evidence is verified by means of the same method used when processing the Joining Response (see Section 4.3.1). Application profiles of this specification MUST specify the exact approaches used by the Client to verify the PoP evidence in 'kdc_cred_verify', and MUST specify which of those approaches is used in which case (REQ21).

Figure 21 gives an overview of the exchange described above, while Figure 22 shows an example.

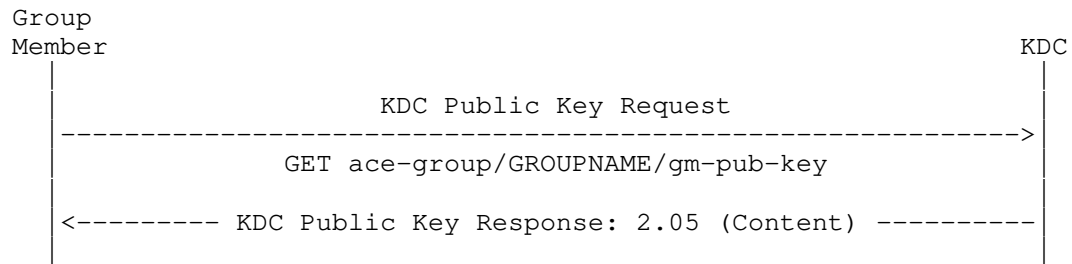


Figure 21: Message Flow of KDC Public Key Request-Response

Request:

```

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "kdc-pub-key"
Payload: -
    
```

Response:

```

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation, with PUB_KEY_KDC
        and POP_EVIDENCE being CBOR byte strings):
{
  "kdc_nonce": h'25a8991cd700ac01',
  "kdc_cred": PUB_KEY_KDC,
  "kdc_cred_verify": POP_EVIDENCE
}
    
```

Figure 22: Example of KDC Public Key Request-Response

4.6. /ace-group/GROUPNAME/policies

This resource implements the GET handler.

4.6.1. GET Handler

The handler expects a GET request.

In addition to what is defined in Section 4.1.2, the handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler replies with a 2.05 (Content) response containing the list of policies for the group identified by GROUPNAME. The payload of the response is formatted as a CBOR map including only the parameter 'group_policies' defined in Section 4.3.1 and specifying the current policies in the group. If the KDC does not store any policy, the payload is formatted as a zero-length CBOR byte string.

The specific format and meaning of group policies MUST be specified in the application profile (REQ20).

4.6.1.1. Retrieve the Group Policies

A node in the group can contact the KDC to retrieve the current group policies, by sending a CoAP GET request to the /ace-group/GROUPNAME/policies endpoint at the KDC, where GROUPNAME is the group name, and formatted as defined in Section 4.6.1

Figure 23 gives an overview of the exchange described above, while Figure 24 shows an example.

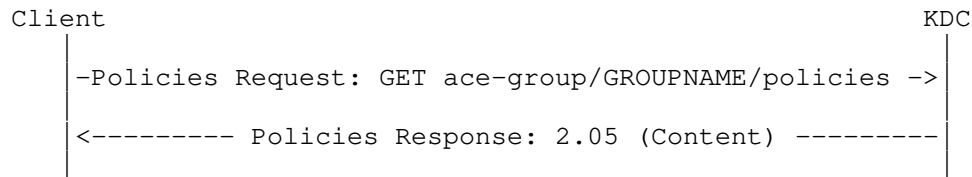


Figure 23: Message Flow of Policies Request-Response

Request:

```
Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "policies"
Payload: -
```

Response:

```
Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload(in CBOR diagnostic notation):
  { "group_policies": {"exp-delta": 120} }
```

Figure 24: Example of Policies Request-Response

4.7. /ace-group/GROUPNAME/num

This resource implements the GET handler.

4.7.1. GET Handler

The handler expects a GET request.

In addition to what is defined in Section 4.1.2, the handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler returns a 2.05 (Content) message containing an integer that represents the version number of the symmetric group keying material. This number is incremented on the KDC every time the KDC updates the symmetric group keying material, before the new keying material is distributed. This number is stored in persistent storage.

The payload of the response is formatted as a CBOR integer.

4.7.1.1. Retrieve the Keying Material Version

A node in the group can contact the KDC to request information about the version number of the symmetric group keying material, by sending a CoAP GET request to the `/ace-group/GROUPNAME/num` endpoint at the KDC, where `GROUENAME` is the group name, formatted as defined in Section 4.7.1. In particular, the version is incremented by the KDC every time the group keying material is renewed, before it's distributed to the group members.

Figure 25 gives an overview of the exchange described above, while Figure 26 shows an example.

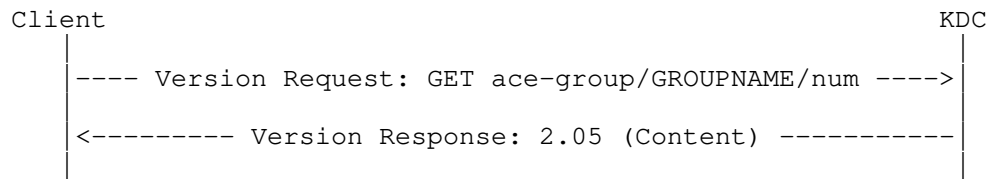


Figure 25: Message Flow of Version Request-Response

Request:

```

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "num"
Payload: -
  
```

Response:

```

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload(in CBOR diagnostic notation):
  13
  
```

Figure 26: Example of Version Request-Response

4.8. `/ace-group/GROUPNAME/nodes/NODENAME`

This resource implements the GET, PUT and DELETE handlers.

In addition to what is defined in Section 4.1.2, each of the handlers performs the following two verifications.

- * The handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 0 ("Operation permitted only to group members").
- * The handler verifies that the node name of the Client is equal to NODENAME used in the url-path. If the verification fails, the handler replies with a 4.03 (Forbidden) error response.

4.8.1. GET Handler

The handler expects a GET request.

If all verifications succeed, the handler replies with a 2.05 (Content) response containing both the group keying material and the individual keying material for the Client, or information enabling the Client to derive it. The payload of the response is formatted as a CBOR map. The format for the group keying material is the same as defined in the response of Section 4.3.2. The specific format of individual keying material for group members, or of the information to derive it, and corresponding CBOR label, MUST be specified in the application profile (REQ27) and registered in Section 11.7.

Optionally, the KDC can make the sub-resource at ace-group/GROUPNAME/nodes/NODENAME also Observable [RFC7641] for the associated node. In case the KDC removes that node from the group without having been explicitly asked for it, this allows the KDC to send an unsolicited 4.04 (Not Found) response to the node as a notification of eviction from the group (see Section 5).

Note that the node could have been observing also the resource at ace-group/GROUPNAME, in order to be informed of changes in the keying material. In such a case, this method would result in largely overlapping notifications received for the resource at ace-group/GROUPNAME and the sub-resource at ace-group/GROUPNAME/nodes/NODENAME.

In order to mitigate this, a node that supports the No-Response option [RFC7967] can use it when starting the observation of the sub-resource at ace-group/GROUPNAME/nodes/NODENAME. In particular, the GET observation request can also include the No-Response option, with value set to 2 (Not interested in 2.xx responses).

4.8.1.1. Retrieve Group and Individual Keying Material

When any of the following happens, a node MUST stop using the owned group keying material to protect outgoing messages, and SHOULD stop using it to decrypt and verify incoming messages.

- * Upon expiration of the keying material, according to what indicated by the KDC with the 'exp' parameter in a Joining Response, or to a pre-configured value.
- * Upon receiving a notification of revoked/renewed keying material from the KDC, possibly as part of an update of the keying material (rekeying) triggered by the KDC.
- * Upon receiving messages from other group members without being able to retrieve the keying material to correctly decrypt them. This may be due to rekeying messages previously sent by the KDC, that the Client was not able to receive or decrypt.

In either case, if it wants to continue participating in the group communication, the node has to request the latest keying material from the KDC. To this end, the Client sends a CoAP GET request to the /ace-group/GROUPNAME/nodes/NODENAME endpoint at the KDC, formatted as specified in Section 4.8.1.

Note that policies can be set up, so that the Client sends a Key Re-Distribution request to the KDC only after a given number of received messages could not be decrypted (because of failed decryption processing or inability to retrieve the necessary keying material).

It is application dependent and pertaining to the particular message exchange (e.g., [I-D.ietf-core-oscure-groupcomm]) to set up these policies for instructing Clients to retain incoming messages and for how long (OPT11). This allows Clients to possibly decrypt such messages after getting updated keying material, rather than just consider them non valid messages to discard right away.

The same Key Distribution Request could also be sent by the Client without being triggered by a failed decryption of a message, if the Client wants to be sure that it has the latest group keying material. If that is the case, the Client will receive from the KDC the same group keying material it already has in memory.

Figure 27 gives an overview of the exchange described above, while Figure 28 shows an example.

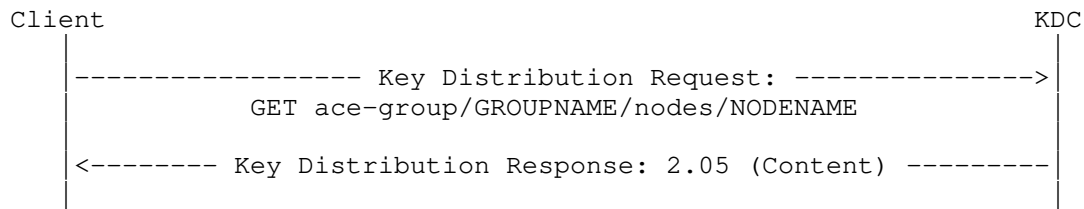


Figure 27: Message Flow of Key Distribution Request-Response

Request:

```

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "nodes"
Uri-Path: "c101"
Payload: -

```

Response:

```

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation,
        with KEY and IND_KEY being CBOR byte strings,
        and "ind-key" the profile-specified label
        for individual keying material):
{ "gkty": 13, "key": KEY, "num": 12, "ind-key": IND_KEY }

```

Figure 28: Example of Key Distribution Request-Response

4.8.2. PUT Handler

The PUT handler processes requests from a Client that asks for new individual keying material, as required to process messages exchanged in the group.

The handler expects a PUT request with empty payload.

In addition to what is defined in Section 4.1.2 and at the beginning of Section 4.8, the handler verifies that this operation is consistent with the set of roles that the Client has in the group (REQ11). If the verification fails, the KDC MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 1 ("Request inconsistent with the current roles").

If the KDC is currently not able to serve this request, i.e., to generate new individual keying material for the requesting Client, the KDC MUST reply with a 5.03 (Service Unavailable) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 4 ("No available node identifiers").

If all verifications succeed, the handler reply with a 2.05 (Content) response containing newly generated, individual keying material for the Client. The payload of the response is formatted as a CBOR map. The specific format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label, MUST be specified in the application profile (REQ27) and registered in Section 11.7.

The typical successful outcome consists in replying with newly generated, individual keying material for the Client, as defined above. However, application profiles of this specification MAY also extend this handler in order to achieve different akin outcomes (OPT12), for instance:

- * Not providing the Client with newly generated, individual keying material, but rather rekeying the whole group, i.e., providing all the current group members with newly generated group keying material.
- * Both providing the Client with newly generated, individual keying material, as well as rekeying the whole group, i.e., providing all the current group members with newly generated group keying material.

In either case, the handler may specify the new group keying material as part of the 2.05 (Content) response.

Note that this handler is not intended to accommodate requests from a group member to trigger a group rekeying, whose scheduling and execution is an exclusive prerogative of the KDC.

4.8.2.1. Request to Change Individual Keying Material

A Client may ask the KDC for new, individual keying material. For instance, this can be due to the expiration of such individual keying material, or to the exhaustion of AEAD nonces, if an AEAD encryption algorithm is used for protecting communications in the group. An example of individual keying material can simply be an individual encryption key associated to the Client. Hence, the Client may ask for a new individual encryption key, or for new input material to derive it.

To this end, the Client performs a Key Renewal Request/Response exchange with the KDC, i.e., it sends a CoAP PUT request to the /ace-group/GROUPNAME/nodes/NODENAME endpoint at the KDC, where GROUPNAME is the group name and NODENAME is its node name, and formatted as defined in Section 4.8.1.

Figure 29 gives an overview of the exchange described above, while Figure 30 shows an example.

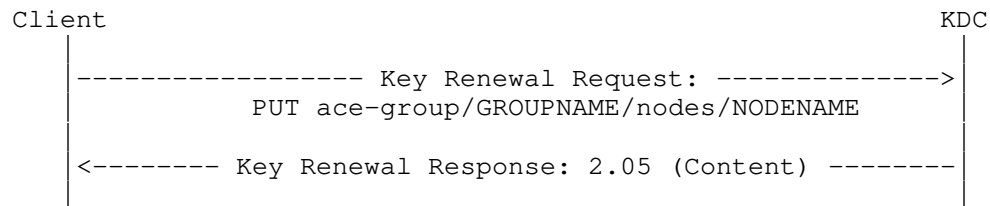


Figure 29: Message Flow of Key Renewal Request-Response

Request:

```

Header: PUT (Code=0.03)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "nodes"
Uri-Path: "c101"
Payload: -
  
```

Response:

```

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation, with IND_KEY being
    a CBOR byte string, and "ind-key" the profile-specified
    label for individual keying material):
{ "ind-key": IND_KEY }
  
```

Figure 30: Example of Key Renewal Request-Response

Note the difference between the Key Renewal Request in this section and the Key Distribution Request in Section 4.8.1.1. The former asks the KDC for new individual keying material, while the latter asks the KDC for the current group keying material together with the current individual keying material.

As discussed in Section 4.8.2, application profiles of this specification may define alternative outcomes for the Key Renewal Request-Response exchange (OPT12), where the provisioning of new individual keying material is replaced by or combined with the execution of a whole group rekeying.

4.8.3. DELETE Handler

The DELETE handler removes the node identified by NODENAME from the group identified by GROUPNAME.

The handler expects a DELETE request with empty payload.

In addition to what is defined in Section 4.1.2, the handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 0 ("Operation permitted only to group members").

If all verification succeeds, the handler performs the actions defined in Section 5 and replies with a 2.02 (Deleted) response with empty payload.

4.8.3.1. Leave the Group

A Client can actively request to leave the group. In this case, the Client sends a CoAP DELETE request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME at the KDC, where GROUPNAME is the group name and NODENAME is its node name, formatted as defined in Section 4.8.3

Note that, after having left the group, the Client may wish to join it again. Then, as long as the Client is still authorized to join the group, i.e., the associated access token is still valid, the Client can request to re-join the group directly to the KDC (see Section 4.3.1.1), without having to retrieve a new access token from the AS.

4.9. /ace-group/GROUPNAME/nodes/NODENAME/pub-key

This resource implements the POST handler.

4.9.1. POST Handler

The POST handler is used to replace the stored public key of this Client (identified by NODENAME) with the one specified in the request at the KDC, for the group identified by GROUPNAME.

The handler expects a POST request with payload as specified in Section 4.3.1, with the difference that it includes only the parameters 'client_cred', 'cnonce' and 'client_cred_verify'. In particular, the PoP evidence included in 'client_cred_verify' is computed in the same way considered in Section 4.3.1 and defined by the specific application profile (REQ14), with a newly generated N_C nonce and the previously received N_S. It is REQUIRED of the application profiles to define the specific formats of public keys that are acceptable to use in the group (REQ6).

In addition to what is defined in Section 4.1.2 and at the beginning of Section 4.8, the handler verifies that this operation is consistent with the set of roles that the node has in the group. If the verification fails, the KDC MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 1 ("Request inconsistent with the current roles").

If the KDC cannot retrieve the 'kdcchallenge' associated to this Client (see Section 3.3), the KDC MUST reply with a 4.00 (Bad Request) error response, which MUST also have Content-Format application/ace-groupcomm+cbor. The payload of the error response is a CBOR map including a newly generated 'kdcchallenge' value. This is specified in the 'kdcchallenge' parameter. In such a case the KDC MUST store the newly generated value as the 'kdcchallenge' value associated to this Client, possibly replacing the currently stored value.

Otherwise, the handler checks that the public key specified in the 'client_cred' field is valid for the group identified by GROUPNAME. That is, the handler checks that the public key is encoded according to the format used in the group, is intended for the public key algorithm used in the group, and is aligned with the possible associated parameters used in the group. If that cannot be successfully verified, the handler MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 2 ("Public key incompatible with the group configuration").

Otherwise, the handler verifies the PoP evidence contained in the 'client_cred_verify' field of the request, by using the public key specified in the 'client_cred' field, as well as the same way considered in Section 4.3.1 and defined by the specific application profile (REQ14). If the PoP evidence does not pass verification, the handler MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 3 ("Invalid Proof-of-Possession evidence").

If all verifications succeed, the handler performs the following actions.

- * The handler associates the public key from the 'client_cred' field of the request to the node identifier NODENAME and to the access token associated to the node identified by NODENAME.
- * In the stored list of group members' public keys for the group identified by GROUPNAME, the handler replaces the public key of the node identified by NODENAME with the public key specified in the 'client_cred' field of the request.

Then, the handler replies with a 2.04 (Changed) response, which does not include a payload.

4.9.1.1. Uploading a New Public Key

In case the KDC maintains the public keys of group members, a node in the group can contact the KDC to upload a new public key to use in the group, and replace the currently stored one.

To this end, the Client performs a Public Key Update Request/Response exchange with the KDC, i.e., it sends a CoAP POST request to the /ace-group/GROUPNAME/nodes/NODENAME/pub-key endpoint at the KDC, where GROUPNAME is the group name and NODENAME is its node name.

The request is formatted as specified in Section 4.9.1.

Figure Figure 31 gives an overview of the exchange described above, while Figure 32 shows an example.

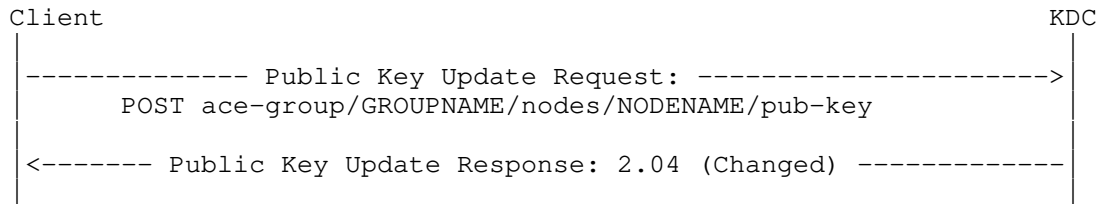


Figure 31: Message Flow of Public Key Update Request-Response

Request:

```

Header: POST (Code=0.02)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "nodes"
Uri-Path: "c101"
Uri-Path: "pub-key"
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation, with PUB_KEY
          and POP_EVIDENCE being CBOR byte strings):
{ "client_cred": PUB_KEY, "nonce": h'9ff7684414affcc8',
  "client_cred_verify": POP_EVIDENCE }
  
```

Response:

```

Header: Changed (Code=2.04)
Payload: -
  
```

Figure 32: Example of Public Key Update Request-Response

Additionally, after updating its own public key, a group member MAY send a number of requests including an identifier of the updated public key, to notify other group members that they have to retrieve it. How this is done depends on the group communication protocol used, and therefore is application profile specific (OPT13).

5. Removal of a Group Member

A Client identified by NODENAME may be removed from a group identified by GROUPNAME where it is a member, due to the following reasons.

1. The Client explicitly asks to leave the group, as defined in Section 4.8.3.1.
2. The node has been found compromised or is suspected so.
3. The Client's authorization to be a group member with the current roles is not valid anymore, i.e., the access token has expired or has been revoked. If the AS provides token introspection (see Section 5.9 of [I-D.ietf-ace-oauth-authz]), the KDC can optionally use it and check whether the Client is still authorized.

In either case, the KDC performs the following actions.

- * The KDC removes the Client from the list of current members or the group.
- * In case of forced eviction, i.e., for cases 2 and 3 above, the KDC deletes the public key of the removed Client, if it acts as repository of public keys for group members.
- * If the removed Client is registered as an observer of the group-membership resource at ace-group/GROUPNAME, the KDC removes the Client from the list of observers of that resource.
- * If the sub-resource nodes/NODENAME was created for the removed Client, the KDC deletes that sub-resource.

In case of forced eviction, i.e., for cases 2 and 3 above, the KDC MAY explicitly inform the removed Client, by means of the following methods.

- If the evicted Client implements the 'control_uri' resource specified in Section 4.3.1, the KDC sends a DELETE request, targeting the URI specified in the 'control_uri' parameter of the Joining Request (see Section 4.3.1).
- If the evicted Client is observing its associated sub-resource at ace-group/GROUPNAME/nodes/NODENAME (see Section 4.8.1), the KDC sends an unsolicited 4.04 (Not Found) error response, which does not include the Observe option and indicates that the observed resource has been deleted (see Section 3.2 of [RFC7641]).

The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 5 ("Group membership terminated").

- * If the application requires forward security or the used application profile requires so, the KDC MUST generate new group keying material and securely distribute it to all the current group members except the leaving node (see Section 6).

6. Group Rekeying Process

A group rekeying is started and driven by the KDC. The KDC is not intended to accommodate explicit requests from group members to trigger a group rekeying. That is, the scheduling and execution of a group rekeying is an exclusive prerogative of the KDC. Reasons that can trigger a group rekeying are a change in the group membership, the current group keying material approaching its expiration time, or a regularly scheduled update of the group keying material.

The KDC MUST increment the version number NUM of the current keying material, before distributing the newly generated keying material with version number NUM+1 to the group. Once completed the group rekeying, the KDC MUST delete the old keying material and SHOULD store the newly distributed keying material in persistent storage.

Distributing the new group keying material requires the KDC to send multiple rekeying messages to the group members. Depending on the rekeying scheme used in the group and the reason that has triggered the rekeying process, each rekeying message can be intended to one or multiple group members, hereafter referred to as target group members. The KDC MUST support at least the "Point-to-Point" group rekeying scheme in Section 6.1 and MAY support additional ones.

Each rekeying message MUST have Content-Format set to application/ace-groupcomm+cbor and its payload formatted as a CBOR map, which MUST include at least the information specified in the Key Distribution Response message (see Section 4.3.2), i.e., the parameters 'gkty', 'key' and 'num' defined in Section 4.3.1. The CBOR map MAY include the parameter 'exp', as well as the parameter 'mgt_key_material' specifying new administrative keying material for the target group members, if relevant for the used rekeying scheme.

A rekeying message may include additional information, depending on the rekeying scheme used in the group, the reason that has triggered the rekeying process and the specific target group members. In particular, if the group rekeying is performed due to one or multiple Clients that have joined the group and the KDC acts as repository of public keys of the group members, then a rekeying message MAY also include the public keys that those Clients use in the group, together with the roles and node identifier that the corresponding Client has in the group. It is RECOMMENDED to specify this information by means of the parameters 'pub_keys', 'peer_roles' and 'peer_identifiers', like done in the Joining Response message (see Section 4.3.1).

The complete format of a rekeying message, including the encoding and content of the 'mgt_key_material' parameter, has to be defined in separate specifications aimed at profiling the used rekeying scheme in the context of the used application profile of this specification. As a particular case, an application profile of this specification MAY define additional information to include in rekeying messages for the "Point-to-Point" group rekeying scheme in Section 6.1 (OPT14).

Consistently with the used group rekeying scheme, the actual delivery of rekeying messages can occur through different approaches, as discussed in the following.

6.1. Point-to-Point Group Rekeying

This approach consists in the KDC sending one individual rekeying message to each target group member. In particular, the rekeying message is protected by means of the security association between the KDC and the target group member in question, as per the used application profile of this specification and the used transport profile of ACE.

This is the approach taken by the basic "Point-to-Point" group rekeying scheme, that the KDC can explicitly signal in the Joining Response (see Section 4.3.1), through the 'rekeying_scheme' parameter specifying the value 0.

When taking this approach in the group identified by GROUPNAME, the KDC can practically deliver the rekeying messages to the target group members in different, co-existing ways.

- * The KDC SHOULD make the ace-group/GROUPNAME resource Observable [RFC7641]. Thus, upon performing a group rekeying, the KDC can distribute the new group keying material through individual notification responses sent to the target group members that are also observing that resource.

In case the KDC deletes the group, this also allows the KDC to send an unsolicited 4.04 (Not Found) response to each observer group member, as a notification of group termination. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4. The value of the 'error' field MUST be set to 6 ("Group deleted").

- * If a target group member specified a URI in the 'control_uri' parameter of the Joining Request upon joining the group (see Section 4.3.1), the KDC can provide that group member with the new group keying material by sending a unicast POST request to that URI.

A Client that does not plan to observe the ace-group/GROUPNAME resource at the KDC SHOULD provide a URI in the 'control_uri' parameter of the Joining Request upon joining the group.

If the KDC has to send a rekeying message to a target group member, but this did not include the 'control_uri' parameter in the Joining Request and is not a registered observer for the ace-group/GROUPNAME resource, then that target group member would not be able to participate to the group rekeying. Later on, after having repeatedly failed to successfully exchange secure messages in the group, that group member can retrieve the current group keying material from the KDC, by sending a GET request to ace-group/GROUPNAME or ace-group/GROUPNAME/nodes/NODENAME (see Section 4.3.2 and Section 4.8.1, respectively).

6.2. One-to-Many Group Rekeying

This section provides high-level recommendations on how the KDC can rekey a group by means of a more efficient and scalable group rekeying scheme, e.g., [RFC2093][RFC2094][RFC2627]. That is, each rekeying message might be, and likely is, intended to multiple target group members, and thus can be delivered to the whole group, although possible to decrypt only for the actual target group members.

This yields an overall lower number of rekeying messages, thus potentially reducing the overall time required to rekey the group. On the other hand, it requires the KDC to provide and use additional administrative keying material to protect the rekeying messages, and to additionally sign them to ensure source authentication (see Section 6.2.1). Typically, this pays off in large-scale groups, where the introduced performance overhead is less than what experienced by rekeying the group in a point-to-point fashion (see Section 6.1).

The exact set of rekeying messages to send, their content and format, the administrative keying material to use to protect them, as well as the set of target group members depend on the specific group rekeying scheme, and are typically affected by the reason that has triggered the group rekeying. Details about the data content and format of rekeying messages have to be defined by separate documents profiling the use of the group rekeying scheme, in the context of the used application profile of this specification.

When one of these group rekeying schemes is used, the KDC provides a number of related information to a Client joining the group in the Joining Response message (see Section 4.3.1). In particular, 'rekeying_scheme' identifies the rekeying scheme used in the group (if no default can be assumed); 'control_group_uri', if present, specifies a URI with a multicast address where the KDC will send the rekeying messages for that group; 'mgt_key_material' specifies a subset of the administrative keying material intended for that particular joining Client to have, as used to protect the rekeying messages sent to the group when intended also to that joining Client.

Rekeying messages can be protected at the application layer, by using COSE and the administrative keying material as prescribed by the specific group rekeying scheme (see Section 6.2.1). After that, the delivery of protected rekeying messages to the intended target group members can occur in different ways, such as the following ones.

- * Over multicast - In this case, the KDC simply sends a rekeying message as a CoAP request addressed to the multicast URI specified in the 'control_group_uri' parameter of the Joining Response (see Section 4.3.1).

If a particular rekeying message is intended to a single target group member, the KDC may alternatively protect the message using the security association with that group member, and deliver the message like when using the "Point-to-Point" group rekeying scheme (see Section 6.1).

- * Through a pub-sub communication model - In this case, the KDC acts as publisher and publishes each rekeying message to a specific "rekeying topic", which is associated to the group and is hosted at a broker server. Following their group joining, the group members subscribe to the rekeying topic at the broker, thus receiving the group rekeying messages as they are published by the KDC.

In order to make such message delivery more efficient, the rekeying topic associated to a group can be further organized into subtopics. For instance, the KDC can use a particular subtopic to

address a particular set of target group members during the rekeying process, as possibly aligned to a similar organization of the administrative keying material (e.g., a key hierarchy).

The setup of rekeying topics at the broker as well as the discovery of the topics at the broker for group members are application specific. A possible way is for the KDC to provide such information in the Joining Response message (see Section 4.3.1), by means of a new parameter analogous to 'control_group_uri' and specifying the URI(s) of the rekeying topic(s) that a group member has to subscribe to at the broker.

Regardless the specifically used delivery method, the group rekeying scheme can perform a possible roll-over of the administrative keying material through the same sent rekeying messages. Actually, such a roll-over occurs every time a group rekeying is performed upon the leaving of group members, which have to be excluded from future communications in the group.

From a high level point of view, each group member owns only a subset of the overall administrative keying material, obtained upon joining the group. Then, when a group rekeying occurs:

- * Each rekeying message is protected by using a (most convenient) key from the administrative keying material such that: i) the used key is not owned by any node leaving the group, i.e. the key is safe to use and does not have to be renewed; and ii) the used key is owned by all the target group members, that indeed have to be provided with new group keying material to protect communications in the group.
- * Each rekeying message includes not only the new group keying material intended to all the rekeyed group members, but also any new administrative keys that: i) are pertaining to and supposed to be owned by the target group members; and ii) had to be updated since leaving group members own the previous version.

Further details depend on the specific rekeying scheme used in the group.

6.2.1. Protection of Rekeying Messages

When using a group rekeying scheme relying on one-to-many rekeying messages, the actual data content of each rekeying message is prepared according to what the rekeying scheme prescribes.

Then, the KDC can protect the rekeying message as defined below. The used encryption algorithm which SHOULD be the same one used to protect communications in the group. The method defined below assumes that the following holds for the management keying material specified in the 'mgt_key_material' parameter of the Joining Response (see Section 4.3.1).

- * The included symmetric encryption keys are accompanied by a corresponding and unique key identifier assigned by the KDC.
- * A Base IV is also included, with the same size of the AEAD nonce considered by the encryption algorithm to use.

First, the KDC computes a COSE_Encrypt0 object as follows.

- * The encryption key to use is selected from the administrative keying material, as defined by the rekeying scheme used in the group.
- * The plaintext is the actual data content of the rekeying message.
- * The Additional Authenticated Data (AAD) is empty, unless otherwise specified by separate documents profiling the use of the group rekeying scheme.
- * Since the KDC is the only sender of rekeying messages, the AEAD nonce can be computed as follows, where NONCE_SIZE is the size in bytes of the AEAD nonce. Separate documents profiling the use of the group rekeying scheme may define alternative ways to compute the AEAD nonce.

The KDC considers the following values.

- COUNT, as a 1-byte unsigned integer associated to the used encryption key. Its value is set to 0 when starting to perform a new group rekeying instance, and is incremented after each use of the encryption key.
- NEW_NUM, as the version number of the new group keying material to distribute in this rekeying instance, left-padded with zeroes to exactly NONCE_SIZE - 1.

Then, the KDC computes a Partial IV as the byte string concatenation of COUNT and NEW_NUM, in this order. Finally, the AEAD nonce is computed as the XOR between the Base IV and the Partial IV.

- * The protected header of the COSE_Encrypt0 object MUST include the following parameters.
 - 'alg', specifying the used encryption algorithm.
 - 'kid', specifying the identifier of the encryption key from the administrative keying material used to protect this rekeying message.
- * The unprotected header of the COSE_Encrypt0 object MUST include the 'Partial IV' parameter, with value the Partial IV computed above.

In order to ensure source authentication, each rekeying message protected with the administrative keying material MUST be signed by the KDC. To this end, the KDC computes a countersignature of the COSE_Encrypt0 object, as described in Sections 3.2 and 3.3 of [I-D.ietf-cose-countersign]. In particular, the following applies when computing the countersignature.

- * The Countersign_structure contains the context text string "CounterSignature0".
- * The private key of the KDC is used as signing key.
- * The payload is the ciphertext of the COSE_Encrypt0 object.
- * The Additional Authenticated Data (AAD) is empty, unless otherwise specified by separate documents profiling the use of a group rekeying scheme.
- * The protected header of the signing object MUST include the parameter 'alg', specifying the used signature algorithm.

If source authentication of messages exchanged in the group is also ensured by means of signatures, then rekeying messages MUST be signed using the same signature algorithm and related parameters. Also, the KDC's public key used for signature verification MUST be provided in the Joining Response through the 'kdc_cred' parameter, together with the corresponding proof-of-possession (PoP) evidence in the 'kdc_cred_verify' parameter.

If source authentication of messages exchanged in the group is not ensured by means of signatures, then the KDC MUST provide its public key together with a corresponding PoP evidence as part of the management keying material specified in the 'mgt_key_material' parameter of the Joining Response (see Section 4.3.1). It is RECOMMENDED to specify this information by using the same format and

encoding used for the parameters 'kdc_cred', 'kdc_nonce' and 'kdc_cred_verify' in the Joining Response. It is up to separate documents profiling the use of the group rekeying scheme to specify such details.

After that, the KDC specifies the computed countersignature in the 'COSE_Countersignature0' header parameter of the COSE_Encrypt0 object.

Finally, the KDC specifies the COSE_Encrypt0 object as payload of a CoAP request, which is sent to the target group members as per the used message delivery method.

7. Extended Scope Format

This section defines an extended format of binary encoded scope, which additionally specifies the semantics used to express the same access control information from the corresponding original scope.

As also discussed in Section 3.2, this enables a Resource Server to unambiguously process a received access token, also in case the Resource Server runs multiple applications or application profiles that involve different scope semantics.

The extended format is intended only for the 'scope' claim of access tokens, for the cases where the claim takes as value a CBOR byte string. That is, the extended format does not apply to the 'scope' parameter included in ACE messages, i.e., the Authorization Request and Authorization Response exchanged between the Client and the Authorization Server (see Sections 5.8.1 and 5.8.2 of [I-D.ietf-ace-oauth-authz]), the AS Request Creation Hints message from the Resource Server (see Section 5.3 of [I-D.ietf-ace-oauth-authz]), and the Introspection Response from the Authorization Server (see Section 5.9.2 of [I-D.ietf-ace-oauth-authz]).

The value of the 'scope' claim following the extended format is composed as follows. Given the original scope using a semantics SEM and encoded as a CBOR byte string, the corresponding extended scope is encoded as a tagged CBOR byte string, wrapping a CBOR sequence [RFC8742] of two elements. In particular:

- * The first element of the sequence is a CBOR integer, and identifies the semantics SEM used for this scope. The value of this element has to be taken from the "Value" column of the "ACE Scope Semantics" registry defined in Section 11.12 of this specification.

When defining a new semantics for a binary scope, it is up to the applications and application profiles to define and register the corresponding integer identifier (REQ28).

- * The second element of the sequence is the original scope using the semantics SEM, encoded as a CBOR byte string.

Finally, the CBOR byte string wrapping the CBOR sequence is tagged, and identified by the CBOR tag TBD_TAG "ACE Extended Scope Format", defined in Section 11.6 of this specification.

The resulting tagged CBOR byte string is used as value of the 'scope' claim of the access token.

The usage of the extended scope format is not limited to application profiles of this specification or to applications based on group communication. Rather, it is generally applicable to any application and application profile where access control information in the access token is expressed as a binary encoded scope.

Figure 33 and Figure 34 build on the examples in Section 3.2, and show the corresponding extended scopes.

```

gname = tstr

permissions = uint . bits roles

roles = &(amp;
    Requester: 1,
    Responder: 2,
    Monitor: 3,
    Verifier: 4
)

scope_entry = AIF_Generic<gname, permissions>

scope = << [ + scope_entry ] >>

semantics = int

; This defines an array, the elements
; of which are to be used in a CBOR Sequence:
sequence = [semantics, scope]

extended_scope = #6.TBD_TAG(<< sequence >>)

```

Figure 33: Example CDLL definition of scope, using the default Authorization Information Format

```

gname = tstr

role = tstr

scope_entry = [ gname , ? ( role / [ 2*role ] ) ]

scope = << [ + scope_entry ] >>

semantics = int

; This defines an array, the elements
; of which are to be used in a CBOR Sequence:
sequence = [semantics, scope]

extended_scope = #6.TBD_TAG(<< sequence >>)

```

Figure 34: CDLL definition of scope, using as example group name encoded as tstr and role as tstr

8. ACE Groupcomm Parameters

This specification defines a number of parameters used during the second part of the message exchange, after the exchange of Token Transfer Request and Response. The table below summarizes them, and specifies the CBOR key to use instead of the full descriptive name.

Note that the media type application/ace-groupcomm+cbor MUST be used when these parameters are transported in the respective message fields.

Name	CBOR Key	CBOR Type	Reference
error	TBD	int	[this document]
error_description	TBD	tstr	[this document]
gid	TBD	array	[this document]
gname	TBD	array of tstr	[this document]
guri	TBD	array of tstr	[this document]
scope	TBD	bstr	[this document]
get_pub_keys	TBD	array / nil	[this document]

client_cred	TBD	bstr	[this document]
cnonce	TBD	bstr	[this document]
client_cred_verify	TBD	bstr	[this document]
pub_keys_repos	TBD	tstr	[this document]
control_uri	TBD	tstr	[this document]
gkty	TBD	int / tstr	[this document]
key	TBD	See the "ACE Groupcomm Key Types" registry	[this document]
num	TBD	int	[this document]
ace-groupcomm-profile	TBD	int	[this document]
exp	TBD	int	[this document]
pub_keys	TBD	array	[this document]
peer_roles	TBD	array	[this document]
peer_identifiers	TBD	array	[this document]
group_policies	TBD	map	[this document]
kdc_cred	TBD	bstr	[this document]
kdc_nonce	TBD	bstr	[this document]
kdc_cred_verify	TBD	bstr	[this document]
rekeying_scheme	TBD	int	[this document]
mgt_key_material	TBD	bstr	[this document]
control_group_uri	TBD	tstr	[this document]
sign_info	TBD	array	[this document]
kdcchallenge	TBD	bstr	[this document]

Figure 35: ACE Groupcomm Parameters

The KDC is expected to support and understand all the parameters above. Instead, a Client can support and understand only a subset of such parameters, depending on the roles it expects to take in the joined groups or on other conditions defined in application profiles of this specification.

In the following, the parameters are categorized according to the support expected by Clients. That is, a Client that supports a parameter is able to: i) use and specify it in a request message to the KDC; and ii) understand and process it if specified in a response message from the KDC. It is REQUIRED of application profiles of this specification to sort their newly defined parameters according to the same categorization (REQ29).

Note that the actual use of a parameter and its inclusion in a message depends on the specific exchange, the specific Client and group involved, as well as what is defined in the used application profile of this specification.

A Client MUST support the following parameters.

- * 'scope', 'gkty', 'key', 'num', 'exp', 'gid', 'gname', 'guri', 'pub_keys', 'peer_identifiers', 'ace_groupcomm_profile', 'control_uri', 'rekeying_scheme'.

A Client SHOULD support the following parameter.

- * 'get_pub_keys'. That is, not supporting this parameter would yield the inconvenient and undesirable behavior where: i) the Client does not ask for the other group members' public keys upon joining the group (see Section 4.3.1.1); and ii) later on as a group member, the Client only retrieves the public keys of all group members (see Section 4.4.2.1).

A Client MAY support the following optional parameters. Application profiles of this specification MAY define that Clients must or should support these parameters instead (OPT15).

- * 'error', 'error_description'.

The following conditional parameters are relevant only if specific conditions hold. It is REQUIRED of application profiles of this specification to define whether Clients must, should or may support these parameters, and under which circumstances (REQ30).

- * 'client_cred', 'cnonce', 'client_cred_verify'. These parameters are relevant for a Client that has a public key to use in a joined group.

- * `'kdcchallenge'`. This parameter is relevant for a Client that has a public key to use in a joined group and that provides the access token to the KDC through a Token Transfer Request (see Section 3.3).
- * `'pub_keys'repo'`. This parameter is relevant for a Client that has a public key to use in a joined group and that makes it available from a key repository different than the KDC.
- * `'group_policies'`. This parameter is relevant for a Client that is interested in the specific policies used in a group, but it does not know them or cannot become aware of them before joining that group.
- * `'peer_roles'`. This parameter is relevant for a Client that has to know about the roles of other group members, especially when retrieving and handling their corresponding public keys.
- * `'kdc_nonce'`, `'kdc_cred'`, `'kdc_cred_verify'`. These parameters are relevant for a Client that joins a group for which, as per the used application profile of this specification, the KDC has an associated public key and this is required for the correct group operation.
- * `'mgt_key_material'`. This parameter is relevant for a Client that supports an advanced rekeying scheme possibly used in the group, such as based on one-to-many rekeying messages sent over IP multicast.
- * `'control_group_uri'`. This parameter is relevant for a Client that supports the hosting of local resources each associated to a group (hence acting as CoAP server) and the reception of one-to-many requests sent to those resources by the KDC (e.g., over IP multicast), targeting multiple members of the corresponding group. Examples of related management operations that the KDC can perform by this means are the eviction of group members and the execution of a group rekeying process through an advanced rekeying scheme, such as based on one-to-many rekeying messages.

9. ACE Groupcomm Error Identifiers

This specification defines a number of values that the KDC can include as error identifiers, in the `'error'` field of an error response with Content-Format `application/ace-groupcomm+cbor`.

Value	Description
0	Operation permitted only to group members
1	Request inconsistent with the current roles
2	Public key incompatible with the group configuration
3	Invalid proof-of-possession evidence
4	No available node identifiers
5	Group membership terminated
6	Group deleted

Figure 36: ACE Groupcomm Error Identifiers

A Client supporting the 'error' parameter (see Section 4.1.2 and Section 8) and able to understand the specified error may use that information to determine what actions to take next. If it is included in the error response and supported by the Client, the 'error_description' parameter may provide additional context.

In particular, the following guidelines apply, and application profiles of this specification can define more detailed actions for the Client to take when learning that a specific error has occurred.

- * In case of error 0, the Client should stop sending the request in question to the KDC. Rather, the Client should first join the targeted group. If it has not happened already, this first requires the Client to obtain an appropriate access token authorizing access to the group and provide it to the KDC.
- * In case of error 1, the Client as a group member should re-join the group with all the roles needed to perform the operation in question. This might require the Client to first obtain a new access token and provide it to the KDC, if the current access token does not authorize to take those roles in the group. For operations admitted to a Client which is not a group member (e.g., an external signature verifier), the Client should first obtain a new access token authorizing to also have the missing roles.

- * In case of error 2, the Client has to obtain or self-generate a different asymmetric key pair, as aligned to the public key algorithms, parameters and encoding used in the targeted group. After that, the Client should provide its new consistent public key to the KDC.
- * In case of error 3, the Client should ensure to be computing its proof-of-possession evidence by correctly using the parameters and procedures defined in the used application profile of this specification. In an unattended setup, it might be not possible for a Client to autonomously diagnose the error and take an effective next action to address it.
- * In case of error 4, the Client should wait for a certain (pre-configured) amount of time, before trying re-sending its request to the KDC.
- * In case of error 5, the Client may try joining the group again. This might require the Client to first obtain a new access token and provide it to the KDC, e.g., if the current access token has expired.
- * In case of error 6, the Client should clean up its state regarding the group, just like if it has left the group with no intention to re-join it.

10. Security Considerations

Security considerations are inherited from the ACE framework [I-D.ietf-ace-oauth-authz], and from the specific transport profile of ACE used between the Clients and the KDC, e.g., [I-D.ietf-ace-dtls-authorize] and [I-D.ietf-ace-oscore-profile].

Furthermore, the following security considerations apply.

10.1. Secure Communication in the Group

When a group member receives a message from a certain sender for the first time since joining the group, it needs to have a mechanism in place to avoid replayed messages, e.g., Appendix B.2 of [RFC8613] or Appendix E of [I-D.ietf-core-oscore-groupcomm]. Such a mechanism aids the recipient group member also in case it has rebooted and lost the security state used to protect previous group communications with that sender.

By its nature, the KDC is invested with a large amount of trust, since it acts as generator and provider of the symmetric keying material used to protect communications in each of its groups. While

details depend on the specific communication and security protocols used in the group, the KDC is in the position to decrypt messages exchanged in the group as if it was also a group member, as long as those are protected through commonly shared group keying material.

A compromised KDC would thus put the attacker in the same position, which also means that:

- * The attacker can generate and control new group keying material, hence possibly rekeying the group and evicting certain group members as part of a broader attack.
- * The attacker can actively participate to communications in a group even without been authorized to join it, and can allow further unauthorized entities to do so.
- * The attacker can build erroneous associations between node identifiers and group members' public keys.

On the other hand, as long as the security protocol used in the group ensures source authentication of messages (e.g., by means of signatures), the KDC is not able to impersonate group members since it does not own their private keys.

Further security considerations are specific of the communication and security protocols used in the group, and thus have to be provided by those protocols and complemented by the application profiles of this specification using them.

10.2. Update of Group Keying Material

Due to different reasons, the KDC can generate new group keying material and provide it to the group members (rekeying) through the rekeying scheme used in the group, as discussed in Section 6.

In particular, the KDC must renew the group keying material latest upon its expiration. Before then, the KDC may also renew the group keying material on a regular or periodical fashion.

The KDC should renew the group keying material upon a group membership change. Since the minimum number of group members is one, the KDC should provide also a Client joining an empty group with new keying material never used before in that group. Similarly, the KDC should provide new group keying material also to a Client that remains the only member in the group after the leaving of other group members.

Note that the considerations in Section 10.1 about dealing with replayed messages still hold, even in case the KDC rekeys the group upon every single joining of a new group member. However, if the KDC has renewed the group keying material upon a group member's joining, and the time interval between the end of the rekeying process and that member's joining is sufficiently small, then that group member is also on the safe side, since it would not accept replayed messages protected with the old group keying material previous to its joining.

The KDC may enforce a rekeying policy that takes into account the overall time required to rekey the group, as well as the expected rate of changes in the group membership. That is, the KDC may not rekey the group at each and every group membership change, for instance if members' joining and leaving occur frequently and performing a group rekeying takes too long. Instead, the KDC might rekey the group after a minimum number of group members have joined or left within a given time interval, or after a maximum amount of time since the last group rekeying was completed, or yet during predictable network inactivity periods.

However, this would result in the KDC not constantly preserving backward and forward security in the group. That is:

- * Newly joining group members would be able to access the keying material used before their joining, and thus they could access past group communications if they have recorded old exchanged messages. This might still be acceptable for some applications and in situations where the new group members are freshly deployed through strictly controlled procedures.
- * The leaving group members would remain able to access upcoming group communications, as protected with the current keying material that has not been updated. This is typically undesirable, especially if the leaving group member is compromised or suspected to be, and it might have an impact or compromise the security properties of the protocols used in the group to protect messages exchanged among the group member.

The KDC should renew the group keying material in case it has rebooted, even in case it stores the whole group keying material in persistent storage. This assumes that the secure associations with the current group members as well as any administrative keying material required to rekey the group are also stored in persistent storage.

However, if the KDC relies on Observe notifications to distribute the new group keying material, the KDC would have lost all the current ongoing Observations with the group members after rebooting, and the

group members would continue using the old group keying material. Therefore, the KDC will rather rely on each group member asking for the new group keying material (see Section 4.3.2.1 and Section 4.8.1.1), or rather perform a group rekeying by actively sending rekeying messages to group members as discussed in Section 6.

The KDC needs to have a mechanism in place to detect DoS attacks from nodes repeatedly performing actions that might trigger a group rekeying. Such actions can include leaving and/or re-joining the group at high rates, or often asking the KDC for new individual keying material. Ultimately, the KDC can resort to removing these nodes from the group and (temporarily) preventing them from joining the group again.

The KDC also needs to have a congestion control mechanism in place, in order to avoid network congestion upon distributing new group keying material. For example, CoAP and Observe give guidance on such mechanisms, see Section 4.7 of [RFC7252] and Section 4.5.1 of [RFC7641].

A node that has left the group should not expect any of its outgoing messages to be successfully processed, if received by other nodes after its leaving, due to a possible group rekeying occurred before the message reception.

10.2.1. Misalignment of Group Keying Material

A group member can receive a message shortly after the group has been rekeyed, and new keying material has been distributed by the KDC (see Section 6). In the following two cases, this may result in misaligned keying material between the group members.

In the first case, the sender protects a message using the old group keying material. However, the recipient receives the message after having received the new group keying material, hence not being able to correctly process it. A possible way to ameliorate this issue is to preserve the old, recent group keying material for a maximum amount of time defined by the application, during which it is used solely for processing incoming messages. By doing so, the recipient can still temporarily process received messages also by using the old, retained group keying material. Note that a former (compromised) group member can take advantage of this by sending messages protected with the old, retained group keying material. Therefore, a conservative application policy should not admit the storage of old group keying material. Eventually, the sender will have obtained the new group keying material too, and can possibly re-send the message protected with such keying material.

In the second case, the sender protects a message using the new group keying material, but the recipient receives that message before having received the new group keying material. Therefore, the recipient would not be able to correctly process the message and hence discards it. If the recipient receives the new group keying material shortly after that and the application at the sender endpoint performs retransmissions, the former will still be able to receive and correctly process the message. In any case, the recipient should actively ask the KDC for the latest group keying material according to an application-defined policy, for instance after a given number of unsuccessfully decrypted incoming messages.

10.3. Block-Wise Considerations

If the Block-Wise CoAP options [RFC7959] are used, and the keying material is updated in the middle of a Block-Wise transfer, the sender of the blocks just changes the group keying material to the updated one and continues the transfer. As long as both sides get the new group keying material, updating group the keying material in the middle of a transfer will not cause any issue. Otherwise, the sender will have to transmit the message again, when receiving an error message from the recipient.

Compared to a scenario where the transfer does not use Block-Wise, depending on how fast the group keying material is changed, the group members might consume a larger amount of the network bandwidth by repeatedly resending the same blocks, which might be problematic.

11. IANA Considerations

This document has the following actions for IANA.

11.1. Media Type Registrations

This specification registers the 'application/ace-groupcomm+cbor' media type for messages of the protocols defined in this document following the ACE exchange and carrying parameters encoded in CBOR. This registration follows the procedures specified in [RFC6838].

Type name: application

Subtype name: ace-groupcomm+cbor

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Must be encoded as CBOR map containing the protocol parameters defined in [this document].

Security considerations: See Section 10 of this document.

Interoperability considerations: n/a

Published specification: [this document]

Applications that use this media type: The type is used by Authorization Servers, Clients and Resource Servers that support the ACE groupcomm framework as specified in [this document].

Fragment identifier considerations: N/A

Additional information: N/A

Person & email address to contact for further information:
iesg@ietf.org (mailto:iesg@ietf.org)

Intended usage: COMMON

Restrictions on usage: None

Author: Francesca Palombini francesca.palombini@ericsson.com
(mailto:francesca.palombini@ericsson.com)

Change controller: IESG

11.2. CoAP Content-Formats

IANA is asked to register the following entry to the "CoAP Content-Formats" registry within the "CoRE Parameters" registry group.

Media Type: application/ace-groupcomm+cbor

Encoding: -

ID: TBD

Reference: [this document]

11.3. OAuth Parameters

IANA is asked to register the following entries in the "OAuth Parameters" registry following the procedure specified in Section 11.2 of [RFC6749].

- * Parameter name: sign_info
- * Parameter usage location: client-rs request, rs-client response
- * Change Controller: IESG
- * Specification Document(s): [[This specification]]

- * Parameter name: kdcchallenge
- * Parameter usage location: rs-client response
- * Change Controller: IESG
- * Specification Document(s): [[This specification]]

11.4. OAuth Parameters CBOR Mappings

IANA is asked to register the following entries in the "OAuth Parameters CBOR Mappings" registry following the procedure specified in Section 8.10 of [I-D.ietf-ace-oauth-authz].

- * Name: sign_info
- * CBOR Key: TBD (range -256 to 255)
- * Value Type: Simple value null / array
- * Reference: [[This specification]]

- * Name: kdcchallenge
- * CBOR Key: TBD (range -256 to 255)
- * Value Type: Byte string
- * Reference: [[This specification]]

11.5. Interface Description (if=) Link Target Attribute Values

IANA is asked to register the following entry in the "Interface Description (if=) Link Target Attribute Values" registry within the "CoRE Parameters" registry group.

- * Attribute Value: ace.group

- * Description: The 'ace group' interface is used to provision keying material and related information and policies to members of a group using the Ace framework.
- * Reference: [This Document]

11.6. CBOR Tags

IANA is asked to register the following entry in the "CBOR Tags" registry.

- * Tag : TBD_TAG
- * Data Item: byte string
- * Semantics: Extended ACE scope format, including the identifier of the used scope semantics.
- * Reference: [This Document]

11.7. ACE Groupcomm Parameters

This specification establishes the "ACE Groupcomm Parameters" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.15.

The columns of this registry are:

- * Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.
- * CBOR Key: This is the value used as CBOR key of the item. These values MUST be unique. The value can be a positive integer, a negative integer, or a string.
- * CBOR Type: This contains the CBOR type of the item, or a pointer to the registry that defines its type, when that depends on another item.
- * Reference: This contains a pointer to the public specification for the item.

This registry has been initially populated by the values in Section 8. The Reference column for all of these entries refers to sections of this document.

11.8. ACE Groupcomm Key Types

This specification establishes the "ACE Groupcomm Key Types" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.15.

The columns of this registry are:

- * Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.
- * Key Type Value: This is the value used to identify the keying material. These values MUST be unique. The value can be a positive integer, a negative integer, or a text string.
- * Profile: This field may contain one or more descriptive strings of application profiles to be used with this item. The values should be taken from the Name column of the "ACE Groupcomm Profiles" registry.
- * Description: This field contains a brief description of the keying material.
- * References: This contains a pointer to the public specification for the format of the keying material, if one exists.

This registry has been initially populated by the values in Figure 10. The specification column for all of these entries will be this document.

11.9. ACE Groupcomm Profiles

This specification establishes the "ACE Groupcomm Profiles" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.15. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

- * Name: The name of the application profile, to be used as value of the profile attribute.

- * Description: Text giving an overview of the application profile and the context it is developed for.
- * CBOR Value: CBOR abbreviation for the name of this application profile. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.
- * Reference: This contains a pointer to the public specification of the abbreviation for this application profile, if one exists.

11.10. ACE Groupcomm Policies

This specification establishes the "ACE Groupcomm Policies" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.15. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

- * Name: The name of the group communication policy.
- * CBOR label: The value to be used to identify this group communication policy. Key map labels MUST be unique. The label can be a positive integer, a negative integer or a string. Integer values between 0 and 255 and strings of length 1 are designated as Standards Track Document required. Integer values from 256 to 65535 and strings of length 2 are designated as Specification Required. Integer values greater than 65535 and strings of length greater than 2 are designated as expert review. Integer values less than -65536 are marked as private use.
- * CBOR type: the CBOR type used to encode the value of this group communication policy.
- * Description: This field contains a brief description for this group communication policy.
- * Reference: This field contains a pointer to the public specification providing the format of the group communication policy, if one exists.

This registry will be initially populated by the values in Figure 11.

11.11. Sequence Number Synchronization Methods

This specification establishes the "Sequence Number Synchronization Methods" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.15. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

- * Name: The name of the sequence number synchronization method.
- * Value: The value to be used to identify this sequence number synchronization method.
- * Description: This field contains a brief description for this sequence number synchronization method.
- * Reference: This field contains a pointer to the public specification describing the sequence number synchronization method.

11.12. ACE Scope Semantics

This specification establishes the "ACE Scope Semantics" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.15. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

- * Value: The value to be used to identify this scope semantics. The value MUST be unique. The value can be a positive integer or a negative integer. Integer values between 0 and 255 are designated as Standards Track Document required. Integer values from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as expert review. Integer values less than -65536 are marked as private use.
- * Description: This field contains a brief description of the scope semantics.

- * Reference: This field contains a pointer to the public specification defining the scope semantics, if one exists.

11.13. ACE Groupcomm Errors

This specification establishes the "ACE Groupcomm Errors" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.15. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

- * Value: The value to be used to identify the error. The value MUST be unique. The value can be a positive integer or a negative integer. Integer values between 0 and 255 are designated as Standards Track Document required. Integer values from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as expert review. Integer values less than -65536 are marked as private use.
- * Description: This field contains a brief description of the error.
- * Reference: This field contains a pointer to the public specification defining the error, if one exists.

This registry has been initially populated by the values in Section 9. The Reference column for all of these entries refers to this document.

11.14. ACE Groupcomm Rekeying Schemes

This specification establishes the "ACE Groupcomm Rekeying Schemes" IANA registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.15. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

- * Value: The value to be used to identify the group rekeying scheme. The value MUST be unique. The value can be a positive integer or a negative integer. Integer values between 0 and 255 are designated as Standards Track Document required. Integer values

from 256 to 65535 are designated as Specification Required.
Integer values greater than 65535 are designated as expert review.
Integer values less than -65536 are marked as private use.

- * Name: The name of the group rekeying scheme.
- * Description: This field contains a brief description of the group rekeying scheme.
- * Reference: This field contains a pointer to the public specification defining the group rekeying scheme, if one exists.

This registry has been initially populated by the value in Figure 12.

11.15. Expert Review Instructions

The IANA Registries established in this document are defined as expert review. This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

- * Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. The zones tagged as private use are intended for testing purposes and closed environments, code points in other ranges should not be assigned for testing.
- * Specifications are required for the standards track range of point assignment. Specifications should exist for specification required ranges, but early assignment before a specification is available is considered to be permissible. Specifications are needed for the first-come, first-serve range if they are expected to be used outside of closed environments in an interoperable way. When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for.

- * Experts should take into account the expected usage of fields when approving point assignment. The fact that there is a range for standards track documents does not mean that a standards track document cannot have points assigned outside of that range. The length of the encoded value should be weighed against how many code points of that length are left, the size of device it will be used on, and the number of code points left that encode to that size.

12. References

12.1. Normative References

- [COSE.Algorithms]
IANA, "COSE Algorithms",
<<https://www.iana.org/assignments/cose/cose.xhtml#algorithms>>.
- [COSE.Header.Parameters]
IANA, "COSE Header Parameters",
<<https://www.iana.org/assignments/cose/cose.xhtml#header-parameters>>.
- [I-D.ietf-ace-aif]
Bormann, C., "An Authorization Information Format (AIF) for ACE", Work in Progress, Internet-Draft, draft-ietf-ace-aif-03, 24 June 2021,
<<https://www.ietf.org/archive/id/draft-ietf-ace-aif-03.txt>>.
- [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)", Work in Progress, Internet-Draft, draft-ietf-ace-oauth-authz-46, 8 November 2021,
<<https://www.ietf.org/archive/id/draft-ietf-ace-oauth-authz-46.txt>>.
- [I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-13, 25 October 2021,
<<https://www.ietf.org/archive/id/draft-ietf-core-oscore-groupcomm-13.txt>>.

- [I-D.ietf-cose-countersign]
Schaad, J. and R. Housley, "CBOR Object Signing and Encryption (COSE): Countersignatures", Work in Progress, Internet-Draft, draft-ietf-cose-countersign-05, 23 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-cose-countersign-05.txt>>.
- [I-D.ietf-cose-rfc8152bis-algs]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-algs-12, 24 September 2020, <<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-algs-12.txt>>.
- [I-D.ietf-cose-rfc8152bis-struct]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-struct-15, 1 February 2021, <<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-struct-15.txt>>.
- [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>>.
- [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>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.
- [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>>.
- [RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <<https://www.rfc-editor.org/info/rfc7967>>.

- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.
- [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>>.
- [RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <<https://www.rfc-editor.org/info/rfc8742>>.
- [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>>.
- [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>>.

12.2. Informative References

- [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)", Work in Progress, Internet-Draft, draft-ietf-ace-dtls-authorize-18, 4 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-ace-dtls-authorize-18.txt>>.
- [I-D.ietf-ace-mqtt-tls-profile]
Sengul, C. and A. Kirby, "Message Queuing Telemetry Transport (MQTT)-TLS profile of Authentication and Authorization for Constrained Environments (ACE) Framework", Work in Progress, Internet-Draft, draft-ietf-ace-mqtt-tls-profile-13, 23 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-ace-mqtt-tls-profile-13.txt>>.

[I-D.ietf-ace-oscore-profile]
Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson,
"OSCORE Profile of the Authentication and Authorization
for Constrained Environments Framework", Work in Progress,
Internet-Draft, draft-ietf-ace-oscore-profile-19, 6 May
2021, <[https://www.ietf.org/archive/id/draft-ietf-ace-
oscore-profile-19.txt](https://www.ietf.org/archive/id/draft-ietf-ace-oscore-profile-19.txt)>.

[I-D.ietf-core-coap-pubsub]
Koster, M., Keranen, A., and J. Jimenez, "Publish-
Subscribe Broker for the Constrained Application Protocol
(CoAP)", Work in Progress, Internet-Draft, draft-ietf-
core-coap-pubsub-09, 30 September 2019,
<[https://www.ietf.org/archive/id/draft-ietf-core-coap-
pubsub-09.txt](https://www.ietf.org/archive/id/draft-ietf-core-coap-pubsub-09.txt)>.

[I-D.ietf-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication
for the Constrained Application Protocol (CoAP)", Work in
Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-
05, 25 October 2021, <[https://www.ietf.org/archive/id/
draft-ietf-core-groupcomm-bis-05.txt](https://www.ietf.org/archive/id/draft-ietf-core-groupcomm-bis-05.txt)>.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsuess, C., and P. V. D. Stok, "Discovery of
OSCORE Groups with the CoRE Resource Directory", Work in
Progress, Internet-Draft, draft-tiloca-core-oscore-
discovery-10, 25 October 2021,
<[https://www.ietf.org/archive/id/draft-tiloca-core-oscore-
discovery-10.txt](https://www.ietf.org/archive/id/draft-tiloca-core-oscore-discovery-10.txt)>.

[RFC2093] Harney, H. and C. Muckenhirn, "Group Key Management
Protocol (GKMP) Specification", RFC 2093,
DOI 10.17487/RFC2093, July 1997,
<<https://www.rfc-editor.org/info/rfc2093>>.

[RFC2094] Harney, H. and C. Muckenhirn, "Group Key Management
Protocol (GKMP) Architecture", RFC 2094,
DOI 10.17487/RFC2094, July 1997,
<<https://www.rfc-editor.org/info/rfc2094>>.

[RFC2627] Wallner, D., Harder, E., and R. Agee, "Key Management for
Multicast: Issues and Architectures", RFC 2627,
DOI 10.17487/RFC2627, June 1999,
<<https://www.rfc-editor.org/info/rfc2627>>.

- [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>>.
- [RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <<https://www.rfc-editor.org/info/rfc7641>>.
- [RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <<https://www.rfc-editor.org/info/rfc7959>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [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>>.
- [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/info/rfc8613>>.

Appendix A. Requirements on Application Profiles

This section lists the requirements on application profiles of this specification, for the convenience of application profile designers.

A.1. Mandatory-to-Address Requirements

- * REQ1: Specify the format and encoding of 'scope'. This includes defining the set of possible roles and their identifiers, as well as the corresponding encoding to use in the scope entries according to the used scope format (see Section 3.1).
- * REQ2: If the AIF format of 'scope' is used, register its specific instance of "Toid" and "Tperm", as well as the corresponding Media Type and Content-Format, as per the guidelines in [I-D.ietf-ace-aif].
- * REQ3: If used, specify the acceptable values for 'sign_alg' (see Section 3.3).

- * REQ4: If used, specify the acceptable values for 'sign_parameters' (see Section 3.3).
- * REQ5: If used, specify the acceptable values for 'sign_key_parameters' (see Section 3.3).
- * REQ6: Specify the acceptable formats for encoding public keys and, if used, the acceptable values for 'pub_key_enc' (see Section 3.3).
- * REQ7: If the value of the GROUPNAME URI path and the group name in the access token scope (gname in Section 3.2) are not required to coincide, specify the mechanism to map the GROUPNAME value in the URI to the group name (see Section 4.1).
- * REQ8: Define whether the KDC has a public key and if this has to be provided through the 'kdc_cred' parameter, see Section 4.3.1.
- * REQ9: Specify if any part of the KDC interface as defined in this document is not supported by the KDC (see Section 4.1).
- * REQ10: Register a Resource Type for the root url-path, which is used to discover the correct url to access at the KDC (see Section 4.1).
- * REQ11: Define what specific actions (e.g., CoAP methods) are allowed on each resource provided by the KDC interface, depending on whether the Client is a current group member; the roles that a Client is authorized to take as per the obtained access token (see Section 3.1); and the roles that the Client has as current group member.
- * REQ12: Categorize possible newly defined operations for Clients into primary operations expected to be minimally supported and secondary operations, and provide accompanying considerations (see Section 4.1.1).
- * REQ13: Specify the encoding of group identifier (see Section 4.2.1).
- * REQ14: Specify the approaches used to compute and verify the PoP evidence to include in 'client_cred_verify', and which of those approaches is used in which case (see Section 4.3.1).
- * REQ15: Specify how the nonce N_S is generated, if the token is not provided to the KDC through the Token Transfer Request to the authz-info endpoint (e.g., if it is used directly to validate TLS instead).

- * REQ16 Define the initial value of the 'num' parameter (see Section 4.3.1).
- * REQ17: Specify the format of the 'key' parameter (see Section 4.3.1).
- * REQ18: Specify the acceptable values of the 'gkty' parameter (see Section 4.3.1).
- * REQ19: Specify and register the application profile identifier (see Section 4.3.1).
- * REQ20: If used, specify the format and content of 'group_policies' and its entries. Specify the policies default values (see Section 4.3.1).
- * REQ21: Specify the approaches used to compute and verify the PoP evidence to include in 'kdc_cred_verify', and which of those approaches is used in which case (see Section 4.3.1).
- * REQ22: Specify the communication protocol the members of the group must use (e.g., multicast CoAP).
- * REQ23: Specify the security protocol the group members must use to protect their communication (e.g., group OSCORE). This must provide encryption, integrity and replay protection.
- * REQ24: Specify how the communication is secured between Client and KDC. Optionally, specify transport profile of ACE [I-D.ietf-ace-oauth-authz] to use between Client and KDC (see Section 4.3.1.1).
- * REQ25: Specify the format of the identifiers of group members (see Section 4.3.1).
- * REQ26: Specify policies at the KDC to handle ids that are not included in 'get_pub_keys' (see Section 4.4.1).
- * REQ27: Specify the format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label (see Section 4.8.1).
- * REQ28: Specify and register the identifier of newly defined semantics for binary scopes (see Section 7).
- * REQ29: Categorize newly defined parameters according to the same criteria of Section 8.

- * REQ30: Define whether Clients must, should or may support the conditional parameters defined in Section 8, and under which circumstances.

A.2. Optional-to-Address Requirements

- * OPT1: Optionally, if the textual format of 'scope' is used, specify CBOR values to use for abbreviating the role identifiers in the group (see Section 3.1).
- * OPT2: Optionally, specify the additional parameters used in the exchange of Token Transfer Request and Response (see Section 3.3).
- * OPT3: Optionally, specify the negotiation of parameter values for signature algorithm and signature keys, if 'sign_info' is not used (see Section 3.3).
- * OPT4: Optionally, specify possible or required payload formats for specific error cases.
- * OPT5: Optionally, specify additional identifiers of error types, as values of the 'error' field in an error response from the KDC.
- * OPT6: Optionally, specify the encoding of 'pub_keys_repos' if the default is not used (see Section 4.3.1).
- * OPT7: Optionally, specify the functionalities implemented at the 'control_uri' resource hosted at the Client, including message exchange encoding and other details (see Section 4.3.1).
- * OPT8: Optionally, specify the behavior of the handler in case of failure to retrieve a public key for the specific node (see Section 4.3.1).
- * OPT9: Optionally, define a default group rekeying scheme, to refer to in case the 'rekeying_scheme' parameter is not included in the Joining Response (see Section 4.3.1).
- * OPT10: Optionally, specify the functionalities implemented at the 'control_group_uri' resource hosted at the Client, including message exchange encoding and other details (see Section 4.3.1).
- * OPT11: Optionally, specify policies that instruct Clients to retain messages and for how long, if they are unsuccessfully decrypted (see Section 4.8.1.1). This makes it possible to decrypt such messages after getting updated keying material.

- * OPT12: Optionally, specify for the KDC to perform group rekeying (together or instead of renewing individual keying material) when receiving a Key Renewal Request (see Section 4.8.2.1).
- * OPT13: Optionally, specify how the identifier of a group members's public key is included in requests sent to other group members (see Section 4.9.1.1).
- * OPT14: Optionally, specify additional information to include in rekeying messages for the "Point-to-Point" group rekeying scheme (see Section 6).
- * OPT15: Optionally, specify if Clients must or should support any of the parameters defined as optional in this specification (see Section 8).

Appendix B. Extensibility for Future COSE Algorithms

As defined in Section 8.1 of [I-D.ietf-cose-rfc8152bis-algs], future algorithms can be registered in the "COSE Algorithms" registry [COSE.Algorithms] as specifying none or multiple COSE capabilities.

To enable the seamless use of such future registered algorithms, this section defines a general, agile format for each 'sign_info_entry' of the 'sign_info' parameter in the Token Transfer Response, see Section 3.3.1.

If any of the currently registered COSE algorithms is considered, using this general format yields the same structure of 'sign_info_entry' defined in this document, thus ensuring retro-compatibility.

B.1. Format of 'sign_info_entry'

The format of each 'sign_info_entry' (see Section 3.3.1) is generalized as follows. Given N the number of elements of the 'sign_parameters' array, i.e., the number of COSE capabilities of the signature algorithm, then:

- * 'sign_key_parameters' is replaced by N elements 'sign_capab_i', each of which is a CBOR array.
- * The i-th array following 'sign_parameters', i.e., 'sign_capab_i' (i = 0, ..., N-1), is the array of COSE capabilities for the algorithm capability specified in 'sign_parameters'[i].

```
sign_info_entry =  
[  
  id : gname / [ + gname ],  
  sign_alg : int / tstr,  
  sign_parameters : [ alg_capab_1 : any,  
                      alg_capab_2 : any,  
                      ...,  
                      alg_capab_N : any],  
  sign_capab_1 : [ any ],  
  sign_capab_2 : [ any ],  
  ...,  
  sign_capab_N : [ any ],  
  pub_key_enc = int / nil  
]  
  
gname = tstr
```

Figure 37: 'sign_info_entry' with general format

Appendix C. Document Updates

RFC EDITOR: PLEASE REMOVE THIS SECTION.

C.1. Version -14 to -15

- * Fixed nits.

C.2. Version -13 to -14

- * Clarified scope and goal of the document in abstract and introduction.
- * Overall clarifications on semantics of operations and parameters.
- * Major restructuring in the presentation of the KDC interface.
- * Revised error handling, also removing redundant text.
- * Imported parameters and KDC resource about the KDC's public key from draft-ietf-ace-key-groupcomm-oscore.
- * New parameters 'group_rekeying_scheme' and 'control_group_uri'.
- * Provided example of administrative keying material transported in 'mgt_key_material'.
- * Reasoned categorization of parameters, as expected support by ACE Clients.

- * Reasoned categorization of KDC functionalities, as minimally/optional to support for ACE Clients.
- * Guidelines on enhanced error responses using 'error' and 'error_description'.
- * New section on group rekeying, discussing at a high-level a basic one-to-one approach and possible one-to-many approaches.
- * Revised and expanded security considerations, also about the KDC.
- * Updated list of requirements for application profiles.
- * Several further clarifications and editorial improvements.

C.3. Version -05 to -13

- * Incremental revision of the KDC interface.
- * Removed redundancy in parameters about signature algorithm and signature keys.
- * Node identifiers always indicated with 'peer_identifiers'.
- * Format of public keys changed from raw COSE Keys to be certificates, CWTs or CWT Claims Set (CCS). Adapted parameter 'pub_key_enc'.
- * Parameters and functionalities imported from draft-ietf-key-groupcomm-oscore where early defined.
- * Possible provisioning of the KDC's Diffie-Hellman public key in response to the Token transferring to /authz-info.
- * Generalized proof-of-possession evidence, to be not necessarily a signature.
- * Public keys of group members may be retrieved filtering by role and/or node identifier.
- * Enhanced error handling with error code and error description.
- * Extended "typed" format for the 'scope' claim, optional to use.
- * Editorial improvements.

C.4. Version -04 to -05

- * Updated uppercase/lowercase URI segments for KDC resources.
- * Supporting single Access Token for multiple groups/topics.
- * Added 'control_uri' parameter in the Joining Request.
- * Added 'peer_roles' parameter to support legal requesters/responders.
- * Clarification on stopping using owned keying material.
- * Clarification on different reasons for processing failures, related policies, and requirement OPT11.
- * Added a KDC sub-resource for group members to upload a new public key.
- * Possible group rekeying following an individual Key Renewal Request.
- * Clarified meaning of requirement REQ3; added requirement OPT12.
- * Editorial improvements.

C.5. Version -03 to -04

- * Revised RESTful interface, as to methods and parameters.
- * Extended processing of joining request, as to check/retrieval of public keys.
- * Revised and extended profile requirements.
- * Clarified specific usage of parameters related to signature algorithms/keys.
- * Included general content previously in draft-ietf-ace-key-groupcomm-oscore
- * Registration of media type and content format application/ace-group+cbor
- * Editorial improvements.

C.6. Version -02 to -03

- * Exchange of information on the signature algorithm and related parameters, during the Token POST (Section 3.3).

- * Restructured KDC interface, with new possible operations (Section 4).
- * Client PoP signature for the Joining Request upon joining (Section 4.1.2.1).
- * Revised text on group member removal (Section 5).
- * Added more profile requirements (Appendix A).

C.7. Version -01 to -02

- * Editorial fixes.
- * Distinction between transport profile and application profile (Section 1.1).
- * New parameters 'sign_info' and 'pub_key_enc' to negotiate parameter values for signature algorithm and signature keys (Section 3.3).
- * New parameter 'type' to distinguish different Key Distribution Request messages (Section 4.1).
- * New parameter 'client_cred_verify' in the Key Distribution Request to convey a Client signature (Section 4.1).
- * Encoding of 'pub_keys_repos' (Section 4.1).
- * Encoding of 'mgt_key_material' (Section 4.1).
- * Improved description on retrieval of new or updated keying material (Section 6).
- * Encoding of 'get_pub_keys' in Public Key Request (Section 7.1).
- * Extended security considerations (Sections 10.1 and 10.2).
- * New "ACE Public Key Encoding" IANA registry (Section 11.2).
- * New "ACE Groupcomm Parameters" IANA registry (Section 11.3), populated with the entries in Section 8.
- * New "Ace Groupcomm Request Type" IANA registry (Section 11.4), populated with the values in Section 9.

- * New "ACE Groupcomm Policy" IANA registry (Section 11.7) populated with two entries "Sequence Number Synchronization Method" and "Key Update Check Interval" (Section 4.2).
- * Improved list of requirements for application profiles (Appendix A).

C.8. Version -00 to -01

- * Changed name of 'req_aud' to 'audience' in the Authorization Request (Section 3.1).
- * Defined error handling on the KDC (Sections 4.2 and 6.2).
- * Updated format of the Key Distribution Response as a whole (Section 4.2).
- * Generalized format of 'pub_keys' in the Key Distribution Response (Section 4.2).
- * Defined format for the message to request leaving the group (Section 5.2).
- * Renewal of individual keying material and methods for group rekeying initiated by the KDC (Section 6).
- * CBOR type for node identifiers in 'get_pub_keys' (Section 7.1).
- * Added section on parameter identifiers and their CBOR keys (Section 8).
- * Added request types for requests to a Join Response (Section 9).
- * Extended security considerations (Section 10).
- * New IANA registries "ACE Groupcomm Key registry", "ACE Groupcomm Profile registry", "ACE Groupcomm Policy registry" and "Sequence Number Synchronization Method registry" (Section 11).
- * Added appendix about requirements for application profiles of ACE on group communication (Appendix A).

Acknowledgments

The following individuals were helpful in shaping this document: Christian Amsuess, Carsten Bormann, Rikard Hoeglund, Ben Kaduk, Watson Ladd, John Mattsson, Daniel Migault, Jim Schaad, Ludwig Seitz, Goeran Selander, Cigdem Sengul and Peter van der Stok.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; by the H2020 project SIFIS-Home (Grant agreement 952652); and by the EIT-Digital High Impact Initiative ACTIVE.

Authors' Addresses

Francesca Palombini
Ericsson AB
Torshamnsgatan 23
SE-16440 Stockholm Kista
Sweden

Email: francesca.palombini@ericsson.com

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Stockholm Kista
Sweden

Email: marco.tiloca@ri.se

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 30 October 2022

M. Tiloca
RISE AB
J. Park
Universitaet Duisburg-Essen
F. Palombini
Ericsson AB
28 April 2022

Key Management for OSCORE Groups in ACE
draft-ietf-ace-key-groupcomm-oscore-14

Abstract

This document defines an application profile of the ACE framework for Authentication and Authorization, to request and provision keying material in group communication scenarios that are based on CoAP and are secured with Group Object Security for Constrained RESTful Environments (Group OSCORE). This application profile delegates the authentication and authorization of Clients, that join an OSCORE group through a Resource Server acting as Group Manager for that group. This application profile leverages protocol-specific transport profiles of ACE to achieve communication security, server authentication and proof-of-possession for a key owned by the Client and bound to an OAuth 2.0 Access Token.

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 30 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	4
1.1. Terminology	5
2. Protocol Overview	6
3. Format of Scope	8
4. Authentication Credentials	10
5. Authorization to Join a Group	12
5.1. Authorization Request	12
5.2. Authorization Response	13
5.3. Token Transferring	13
5.3.1. 'ecdh_info' Parameter	16
5.3.2. 'kdc_dh_creds' Parameter	18
6. Group Joining	20
6.1. Send the Joining Request	20
6.1.1. Value of the N_S Challenge	22
6.2. Receive the Joining Request	22
6.2.1. Follow-up to a 4.00 (Bad Request) Error Response	25
6.3. Send the Joining Response	26
6.4. Receive the Joining Response	32
7. Overview of the Group Rekeying Process	34
7.1. Stale OSCORE Sender IDs	35
8. Interface at the Group Manager	37
8.1. ace-group/GROUPNAME/active	37
8.1.1. GET Handler	37
8.2. ace-group/GROUPNAME/verif-data	38
8.2.1. GET Handler	38
8.3. ace-group/GROUPNAME/stale-sids	38
8.3.1. FETCH Handler	38
8.4. Admitted Methods	39
8.4.1. Signature Verifiers	40
8.5. Operations Supported by Clients	41
9. Additional Interactions with the Group Manager	41
9.1. Retrieve Updated Keying Material	42
9.1.1. Get Group Keying Material	42
9.1.2. Get Group Keying Material and OSCORE Sender ID	42
9.2. Request to Change Individual Keying Material	43
9.3. Retrieve Authentication Credentials of Group Members	45
9.4. Upload a New Authentication Credential	45

9.5.	Retrieve the Group Manager's Authentication Credential	47
9.6.	Retrieve Signature Verification Data	48
9.7.	Retrieve the Group Policies	50
9.8.	Retrieve the Keying Material Version	50
9.9.	Retrieve the Group Status	50
9.10.	Retrieve Group Names	51
9.11.	Leave the Group	54
10.	Removal of a Group Member	54
11.	Group Rekeying Process	56
11.1.	Sending Rekeying Messages	58
11.2.	Receiving Rekeying Messages	60
11.3.	Missed Rekeying Instances	61
11.3.1.	Retrieve Stale Sender IDs	63
12.	ACE Groupcomm Parameters	65
13.	ACE Groupcomm Error Identifiers	67
14.	Default Values for Group Configuration Parameters	68
14.1.	Common	68
14.2.	Group Mode	69
14.3.	Pairwise Mode	70
15.	Security Considerations	71
15.1.	Management of OSCORE Groups	71
15.2.	Size of Nonces as Proof-of-Possession Challenge	72
15.3.	Reusage of Nonces for Proof-of-Possession Input	73
16.	IANA Considerations	74
16.1.	OAuth Parameters	74
16.2.	OAuth Parameters CBOR Mappings	74
16.3.	ACE Groupcomm Parameters	75
16.4.	ACE Groupcomm Key Types	76
16.5.	ACE Groupcomm Profiles	76
16.6.	OSCORE Security Context Parameters	76
16.7.	TLS Exporter Labels	78
16.8.	AIF	79
16.9.	CoAP Content-Format	79
16.10.	Group OSCORE Roles	80
16.11.	CoRE Resource Type	80
16.12.	ACE Scope Semantics	81
16.13.	ACE Groupcomm Errors	81
16.14.	Expert Review Instructions	82
17.	References	82
17.1.	Normative References	82
17.2.	Informative References	86
Appendix A.	Profile Requirements	88
A.1.	Mandatory-to-Address Requirements	88
A.2.	Optional-to-Address Requirements	91
Appendix B.	Extensibility for Future COSE Algorithms	92
B.1.	Format of 'ecdh_info_entry'	93
B.2.	Format of 'key'	94
Appendix C.	Document Updates	95

C.1.	Version -13 to -14	95
C.2.	Version -12 to -13	95
C.3.	Version -11 to -12	95
C.4.	Version -10 to -11	96
C.5.	Version -09 to -10	97
C.6.	Version -08 to -09	97
C.7.	Version -07 to -08	98
C.8.	Version -06 to -07	98
C.9.	Version -05 to -06	99
C.10.	Version -04 to -05	99
C.11.	Version -03 to -04	100
C.12.	Version -02 to -03	100
C.13.	Version -01 to -02	101
C.14.	Version -00 to -01	102
Acknowledgments		102
Authors' Addresses		102

1. Introduction

Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] is a method for application-layer protection of the Constrained Application Protocol (CoAP) [RFC7252], using CBOR Object Signing and Encryption (COSE) [I-D.ietf-cose-rfc8152bis-struct][I-D.ietf-cose-rfc8152bis-algs] and enabling end-to-end security of CoAP payload and options.

As described in [I-D.ietf-core-oscore-groupcomm], Group OSCORE is used to protect CoAP group communication [I-D.ietf-core-groupcomm-bis], which can employ, for example, IP multicast as underlying data transport. This relies on a Group Manager, which is responsible for managing an OSCORE group and enables the group members to exchange CoAP messages secured with Group OSCORE. The Group Manager can be responsible for multiple groups, coordinates the joining process of new group members, and is entrusted with the distribution and renewal of group keying material.

This document is an application profile of [I-D.ietf-ace-key-groupcomm], which itself builds on the ACE framework for Authentication and Authorization [I-D.ietf-ace-oauth-authz]. Message exchanges among the participants as well as message formats and processing follow what specified in [I-D.ietf-ace-key-groupcomm] for provisioning and renewing keying material in group communication scenarios, where Group OSCORE is used to protect CoAP group communication.

1.1. Terminology

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

Readers are expected to be familiar with:

- * The terms and concepts described in the ACE framework for authentication and authorization [I-D.ietf-ace-oauth-authz] and in the Authorization Information Format (AIF) [I-D.ietf-ace-aif] to express authorization information. The terminology for entities in the considered architecture is defined in OAuth 2.0 [RFC6749]. In particular, this includes Client (C), Resource Server (RS), and Authorization Server (AS).
- * The terms and concept related to the message formats and processing specified in [I-D.ietf-ace-key-groupcomm], for provisioning and renewing keying material in group communication scenarios.
- * The terms and concepts described in CBOR [RFC8949] and COSE [I-D.ietf-cose-rfc8152bis-struct][I-D.ietf-cose-rfc8152bis-algs].
- * The terms and concepts described in CoAP [RFC7252] and group communication for CoAP [I-D.ietf-core-groupcomm-bis]. Unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol".
- * The terms and concepts for protection and processing of CoAP messages through OSCORE [RFC8613] and through Group OSCORE [I-D.ietf-core-oscore-groupcomm] in group communication scenarios. These especially include:
 - Group Manager, as the entity responsible for a set of groups where communications are secured with Group OSCORE. In this document, the Group Manager acts as Resource Server.

- Authentication credential, as the set of information associated with an entity, including that entity's public key and parameters associated with the public key. Examples of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC7925] and C509 certificates [I-D.ietf-cose-cbor-encoded-cert].

Additionally, this document makes use of the following terminology.

- * Requester: member of an OSCORE group that sends request messages to other members of the group.
- * Responder: member of an OSCORE group that receives request messages from other members of the group. A responder may reply back, by sending a response message to the requester which has sent the request message.
- * Monitor: member of an OSCORE group that is configured as responder and never replies back to requesters after receiving request messages. This corresponds to the term "silent server" used in [I-D.ietf-core-oscore-groupcomm].
- * Signature verifier: entity external to the OSCORE group and intended to verify the signature of messages exchanged in the group (see Sections 3.1 and 8.5 of [I-D.ietf-core-oscore-groupcomm]). An authorized signature verifier does not join the OSCORE group as an actual member, yet it can retrieve the authentication credentials of the current group members from the Group Manager.
- * Signature-only group: an OSCORE group that uses only the group mode (see Section 8 of [I-D.ietf-core-oscore-groupcomm]).
- * Pairwise-only group: an OSCORE group that uses only the pairwise mode (see Section 9 of [I-D.ietf-core-oscore-groupcomm]).

2. Protocol Overview

Group communication for CoAP has been enabled in [I-D.ietf-core-groupcomm-bis] and can be secured with Group Object Security for Constrained RESTful Environments (Group OSCORE) as specified in [I-D.ietf-core-oscore-groupcomm]. A network node joins an OSCORE group by interacting with the responsible Group Manager. Once registered in the group, the new node can securely exchange messages with other group members.

This document describes how to use [I-D.ietf-ace-key-groupcomm] and [I-D.ietf-ace-oauth-authz] to perform a number of authentication, authorization and key distribution actions as overviewed in Section 2 of [I-D.ietf-ace-key-groupcomm], when the considered group is specifically an OSCORE group.

With reference to [I-D.ietf-ace-key-groupcomm]:

- * The node wishing to join the OSCORE group, i.e., the joining node, is the Client.
- * The Group Manager is the Key Distribution Center (KDC), acting as a Resource Server.
- * The Authorization Server associated with the Group Manager is the AS.

A node performs the steps described in Sections 3 and 4.3.1.1 of [I-D.ietf-ace-key-groupcomm] in order to obtain an authorization for joining an OSCORE group and then to join that group. The format and processing of messages exchanged during such steps are further specified in Section 5 and Section 6 of this document.

All communications between the involved entities MUST be secured.

In particular, communications between the Client and the Group Manager leverage protocol-specific transport profiles of ACE to achieve communication security, proof-of-possession and server authentication. It is expected that, in the commonly referred base-case of this document, the transport profile to use is pre-configured and well-known to nodes participating in constrained applications.

With respect to what is defined in [I-D.ietf-ace-key-groupcomm]:

- * The interface provided by the Group Manager extends the original interface defined in Section 4.1 of [I-D.ietf-ace-key-groupcomm] for the KDC, as specified in Section 8 of this document.
- * In addition to those defined in Section 8 of [I-D.ietf-ace-key-groupcomm], additional parameters are defined in this document and summarized in Section 12.
- * In addition to those defined in Section 9 of [I-D.ietf-ace-key-groupcomm], additional error identifiers are defined in this document and summarized in Section 13.

Finally, Appendix A lists the specifications on this application profile of ACE, based on the requirements defined in Appendix A of [I-D.ietf-ace-key-groupcomm].

3. Format of Scope

Building on Section 3.1 of [I-D.ietf-ace-key-groupcomm], this section defines the exact format and encoding of scope used in this profile.

To this end, this profile uses the Authorization Information Format (AIF) [I-D.ietf-ace-aif]. In particular, with reference to the generic AIF model

AIF-Generic<Toid, Tperm> = [* [Toid, Tperm]]

the value of the CBOR byte string used as scope encodes the CBOR array [* [Toid, Tperm]], where each [Toid, Tperm] element corresponds to one scope entry.

Furthermore, this document defines the new AIF specific data model AIF-OSCORE-GROUPCOMM, that this profile MUST use to format and encode scope entries.

In particular, the following holds for each scope entry.

- * The object identifier ("Toid") is specialized as a CBOR item specifying the name of the groups pertaining to the scope entry.
- * The permission set ("Tperm") is specialized as a CBOR unsigned integer with value R, specifying the permissions that the Client wishes to have in the groups indicated by "Toid".

More specifically, the following applies when, as defined in this document, a scope entry includes as set of permissions the set of roles to take in an OSCORE group.

- * The object identifier ("Toid") is a CBOR text string, specifying the group name for the scope entry.
- * The permission set ("Tperm") is a CBOR unsigned integer with value R, specifying the role(s) that the Client wishes to take in the group (REQ1). The value R is computed as follows.
 - Each role in the permission set is converted into the corresponding numeric identifier X from the "Value" column of the "Group OSCORE Roles" registry, for which this document defines the entries in Figure 1.

- The set of N numbers is converted into the single value R, by taking two to the power of each numeric identifier X₁, X₂, ..., X_N, and then computing the inclusive OR of the binary representations of all the power values.

Name	Value	Description
Reserved	0	This value is reserved
Requester	1	Send requests; receive responses
Responder	2	Send responses; receive requests
Monitor	3	Receive requests; never send requests/responses
Verifier	4	Verify signature of intercepted messages

Figure 1: Numeric identifier of roles in an OSCORE group

The following CDDL [RFC8610] notation defines a scope entry that uses the AIF-OSCORE-GROUPCOMM data model and expresses a set of Group OSCORE roles from those in Figure 1.

```

AIF-OSCORE-GROUPCOMM = AIF-Generic<oscore-gname, oscore-gperm>

oscore-gname = tstr ; Group name
oscore-gperm = uint . bits group-oscore-roles

group-oscore-roles = &(
    Requester: 1,
    Responder: 2,
    Monitor: 3,
    Verifier: 4
)

scope_entry = [oscore-gname, oscore-gperm]
```

Future specifications that define new Group OSCORE roles MUST register a corresponding numeric identifier in the "Group OSCORE Roles" registry defined in Section 16.10 of this document.

Note that the value 0 is not available to use as numeric identifier to specify a Group OSCORE role. It follows that, when expressing Group OSCORE roles to take in a group as per this document, a scope entry has the least significant bit of "Tperm" always set to 0.

This is an explicit feature of the AIF-OSCORE-GROUPCOMM data model. That is, for each scope entry, the least significant bit of "Tperm" set to 0 explicitly identifies the scope entry as exactly expressing a set of Group OSCORE roles ("Tperm"), pertaining to a single group whose name is specified by the string literal in "Toid".

Instead, by relying on the same AIF-OSCORE-GROUPCOMM data model, [I-D.ietf-ace-oscore-gm-admin] defines the format of scope entries for Administrator Clients that wish to access an admin interface at the Group Manager. In such scope entries, the least significant bit of "Tperm" is always set to 1.

4. Authentication Credentials

Source authentication of a message sent within the group and protected with Group OSCORE is ensured by means of a digital signature embedded in the message (in group mode), or by integrity-protecting the message with pairwise keying material derived from the asymmetric keys of sender and recipient (in pairwise mode).

Therefore, group members must be able to retrieve each other's authentication credential from a trusted repository, in order to verify source authenticity of incoming group messages.

As also discussed in [I-D.ietf-core-oscore-groupcomm], the Group Manager acts as trusted repository of the authentication credentials of the group members, and provides those authentication credentials to group members if requested to. Upon joining an OSCORE group, a joining node is thus expected to provide its own authentication credential to the Group Manager.

In particular, one of the following four cases can occur when a new node joins an OSCORE group.

- * The joining node is going to join the group exclusively as monitor, i.e., it is not going to send messages to the group. In this case, the joining node is not required to provide its own authentication credential to the Group Manager, which thus does not have to perform any check related to the format of the authentication credential, to a signature or ECDH algorithm, and to possible parameters associated with the algorithm and the public key. In case the joining node still provides an authentication credential in the 'client_cred' parameter of the Joining Request (see Section 6.1), the Group Manager silently ignores that parameter, as well as the related parameters 'cnonce' and 'client_cred_verify'.

- * The Group Manager already acquired the authentication credential of the joining node during a past joining process. In this case, the joining node MAY choose not to provide again its own authentication credential to the Group Manager, in order to limit the size of the Joining Request. The joining node MUST provide its own authentication credential again if it has provided the Group Manager with multiple authentication credentials during past joining processes, intended for different OSCORE groups. If the joining node provides its own authentication credential, the Group Manager performs consistency checks as per Section 6.2 and, in case of success, considers it as the authentication credential associated with the joining node in the OSCORE group.
- * The joining node and the Group Manager use an asymmetric proof-of-possession key to establish a secure communication association. Then, two cases can occur.
 1. When establishing the secure communication association, the Group Manager obtained from the joining node the joining node's authentication credential, in the format used in the OSCORE group and including the asymmetric proof-of-possession key as public key. Also, such authentication credential and the proof-of-possession key are compatible with the signature or ECDH algorithm, and possible associated parameters used in the OSCORE group.

In this case, the Group Manager considers the authentication credential as the one associated with the joining node in the OSCORE group. If the joining node is aware that the authentication credential and the public key included thereof are also valid for the OSCORE group, then the joining node MAY choose to not provide again its own authentication credential to the Group Manager.

The joining node MUST provide again its own authentication credential if it has provided the Group Manager with multiple authentication credentials during past joining processes, intended for different OSCORE groups. If the joining node provides its own authentication credential in the 'client_cred' parameter of the Joining Request (see Section 6.1), the Group Manager performs consistency checks as per Section 6.2 and, in case of success, considers it as the authentication credential associated with the joining node in the OSCORE group.

2. The authentication credential is not in the format used in the OSCORE group, or else the authentication credential and the proof-of-possession key included as public key are not compatible with the signature or ECDH algorithm, and possible associated parameters used in the OSCORE group.

In this case, the joining node MUST provide a different compatible authentication credential and public key included thereof to the Group Manager in the 'client_cred' parameter of the Joining Request (see Section 6.1). Then, the Group Manager performs consistency checks on this latest provided authentication credential as per Section 6.2 and, in case of success, considers it as the authentication credential associated with the joining node in the OSCORE group.

- * The joining node and the Group Manager use a symmetric proof-of-possession key to establish a secure communication association. In this case, upon performing a joining process with that Group Manager for the first time, the joining node specifies its own authentication credential in the 'client_cred' parameter of the Joining Request (see Section 6.1).

5. Authorization to Join a Group

This section builds on Section 3 of [I-D.ietf-ace-key-groupcomm] and is organized as follows.

First, Section 5.1 and Section 5.2 describe how the joining node interacts with the AS, in order to be authorized to join an OSCORE group under a given Group Manager and to obtain an Access Token. Then, Section 5.3 describes how the joining node transfers the obtained Access Token to the Group Manager. The following considers a joining node that intends to contact the Group Manager for the first time.

Note that what is defined in Section 3 of [I-D.ietf-ace-key-groupcomm] applies, and only additions or modifications to that specification are defined in this document.

5.1. Authorization Request

The Authorization Request message is as defined in Section 3.1 of [I-D.ietf-ace-key-groupcomm], with the following additions.

- * If the 'scope' parameter is present:

- The value of the CBOR byte string encodes a CBOR array, whose format MUST follow the data model AIF-OSCORE-GROUPCOMM defined in Section 3. In particular, for each OSCORE group to join:
 - o The group name is encoded as a CBOR text string.
 - o The set of requested roles is expressed as a single CBOR unsigned integer. This is computed as defined in Section 3, from the numerical abbreviations of each requested role defined in the "Group OSCORE Roles" registry, for which this document defines the entries in Figure 1 (REQ1).

5.2. Authorization Response

The Authorization Response message is as defined in Section 3.2 of [I-D.ietf-ace-key-groupcomm], with the following additions:

- * The AS MUST include the 'expires_in' parameter. Other means for the AS to specify the lifetime of Access Tokens are out of the scope of this document.
- * The AS MUST include the 'scope' parameter, when the value included in the Access Token differs from the one specified by the joining node in the Authorization Request. In such a case, the second element of each scope entry MUST be present, and specifies the set of roles that the joining node is actually authorized to take in the OSCORE group for that scope entry, encoded as specified in Section 5.1.

Furthermore, if the AS uses the extended format of scope defined in Section 7 of [I-D.ietf-ace-key-groupcomm] for the 'scope' claim of the Access Token, the first element of the CBOR sequence [RFC8742] MUST be the CBOR integer with value SEM_ID_TBD, defined in Section 16.12 of this document (REQ28). This indicates that the second element of the CBOR sequence, as conveying the actual access control information, follows the scope semantics defined for this application profile in Section 3 of this document.

5.3. Token Transferring

The exchange of Token Transfer Request and Token Transfer Response is defined in Section 3.3 of [I-D.ietf-ace-key-groupcomm]. In addition to that, the following applies.

- * The Token Transfer Request MAY additionally contain the following parameters, which, if included, MUST have the corresponding values defined below (OPT2):

- 'ecdh_info' defined in Section 5.3.1 of this document, with value the CBOR simple value "null" (0xf6) to request information about the ECDH algorithm, the ECDH algorithm parameters, the ECDH key parameters and the exact format of authentication credentials used in the groups that the Client has been authorized to join. This is relevant in case the joining node supports the pairwise mode of Group OSCORE [I-D.ietf-core-oscore-groupcomm].
- 'kdc_dh_creds' defined in Section 5.3.2 of this document, with value the CBOR simple value "null" (0xf6) to request the Diffie-Hellman authentication credentials of the Group Manager for the groups that the Client has been authorized to join. That is, each of such authentication credentials includes a Diffie-Hellman public key of the Group Manager. This is relevant in case the joining node supports the pairwise mode of Group OSCORE [I-D.ietf-core-oscore-groupcomm].

Alternatively, the joining node may retrieve this information by other means.

- * The 'kdcchallenge' parameter contains a dedicated nonce N_S generated by the Group Manager. For the N_S value, it is RECOMMENDED to use a 8-byte long random nonce. The joining node can use this nonce in order to prove the possession of its own private key, upon joining the group (see Section 6.1).

The 'kdcchallenge' parameter MAY be omitted from the Token Transfer Response, if the 'scope' of the Access Token specifies only the role "monitor" or only the role "verifier" or only the two roles combined, for each and every of the specified groups.

- * If the 'sign_info' parameter is present in the response, the following applies for each element 'sign_info_entry'.
 - 'id' MUST NOT refer to OSCORE groups that are pairwise-only groups.
 - 'sign_alg' takes value from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].
 - 'sign_parameters' is a CBOR array. Its format and value are the same of the COSE capabilities array for the algorithm indicated in 'sign_alg', as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms] (REQ4).

- 'sign_key_parameters' is a CBOR array. Its format and value are the same of the COSE capabilities array for the COSE key type of the keys used with the algorithm indicated in 'sign_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types] (REQ5).
- 'pub_key_enc' takes value from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters] (REQ6). Consistently with Section 2.3 of [I-D.ietf-core-oscore-groupcomm], acceptable values denote a format of authentication credential that MUST explicitly provide the public key as well as the comprehensive set of information related to the public key algorithm, including, e.g., the used elliptic curve (when applicable).

At the time of writing this specification, acceptable formats of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC7925] and C509 certificates [I-D.ietf-cose-cbor-encoded-cert]. Further formats may be available in the future, and would be acceptable to use as long as they comply with the criteria defined above.

[As to CWTs and CCSs, the COSE Header Parameters 'kcwt' and 'kccs' are under pending registration requested by draft-ietf-lake-edhoc.]

[As to C509 certificates, the COSE Header Parameters 'c5b' and 'c5c' are under pending registration requested by draft-ietf-cose-cbor-encoded-cert.]

This format is consistent with every signature algorithm currently considered in [I-D.ietf-cose-rfc8152bis-algs], i.e., with algorithms that have only the COSE key type as their COSE capability. Appendix B of [I-D.ietf-ace-key-groupcomm] describes how the format of each 'sign_info_entry' can be generalized for possible future registered algorithms having a different set of COSE capabilities.

- * If 'ecdh_info' is included in the Token Transfer Request, the Group Manager SHOULD include the 'ecdh_info' parameter in the Token Transfer Response, as per the format defined in Section 5.3.1. Note that the field 'id' of each 'ecdh_info_entry' specifies the name, or array of group names, for which that 'ecdh_info_entry' applies to.

As an exception, the KDC MAY omit the 'ecdh_info' parameter in the Token Transfer Response even if 'ecdh_info' is included in the Token Transfer Request, in case all the groups that the Client is authorized to join are signature-only groups.

- * If 'kdc_dh_creds' is included in the Token Transfer Request and any of the groups that the Client has been authorized to join is a pairwise-only group, then the Group Manager MUST include the 'kdc_dh_creds' parameter in the Token Transfer Response, as per the format defined in Section 5.3.2. Otherwise, if 'kdc_dh_creds' is included in the Token Transfer Request, the Group Manager MAY include the 'kdc_dh_creds' parameter in the Token Transfer Response. Note that the field 'id' specifies the group name, or array of group names, for which the corresponding 'kdc_dh_creds' applies to.

Note that, other than through the above parameters as defined in Section 3.3 of [I-D.ietf-ace-key-groupcomm], the joining node may have obtained such information by alternative means. For example, information conveyed in the 'sign_info' and 'ecdh_info' parameters may have been pre-configured, or the joining node MAY early retrieve it by using the approach described in [I-D.tiloca-core-oscore-discovery], to discover the OSCORE group and the link to the associated group-membership resource at the Group Manager (OPT3).

5.3.1. 'ecdh_info' Parameter

The 'ecdh_info' parameter is an OPTIONAL parameter of the request and response messages exchanged between the Client and the authz-info endpoint at the RS (see Section 5.10.1. of [I-D.ietf-ace-oauth-authz]).

This parameter allows the Client and the RS to exchange information about an ECDH algorithm as well as about the authentication credentials and public keys to accordingly use for deriving Diffie-Hellman secrets. Its exact semantics and content are application specific.

In this application profile, this parameter is used to exchange information about the ECDH algorithm as well as about the authentication credentials and public keys to be used with it, in the groups indicated by the transferred Access Token as per its 'scope' claim (see Section 3.2 of [I-D.ietf-ace-key-groupcomm]).

When used in the Token Transfer Request sent to the Group Manager, the 'ecdh_info' parameter has value the CBOR simple value "null" (0xf6). This is done to ask for information about the ECDH algorithm

as well as about the authentication credentials and public keys to be used to compute static-static Diffie-Hellman shared secrets [NIST-800-56A], in the OSCORE groups that the Client has been authorized to join and that use the pairwise mode of Group OSCORE [I-D.ietf-core-oscore-groupcomm].

When used in the following Token Transfer Response from the Group Manager, the 'ecdh_info' parameter is a CBOR array of one or more elements. The number of elements is at most the number of OSCORE groups that the Client has been authorized to join.

Each element contains information about ECDH parameters as well as about authentication credentials and public keys, for one or more OSCORE groups that use the pairwise mode of Group OSCORE and that the Client has been authorized to join. Each element is formatted as follows.

- * The first element 'id' is the group name of the OSCORE group or an array of group names for the OSCORE groups for which the specified information applies. In particular 'id' MUST NOT refer to OSCORE groups that are signature-only groups.
- * The second element 'ecdh_alg' is a CBOR integer or a CBOR text string indicating the ECDH algorithm used in the OSCORE group identified by 'gname'. Values are taken from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].
- * The third element 'ecdh_parameters' is a CBOR array indicating the parameters of the ECDH algorithm used in the OSCORE group identified by 'gname'. Its format and value are the same of the COSE capabilities array for the algorithm indicated in 'ecdh_alg', as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms].
- * The fourth element 'ecdh_key_parameters' is a CBOR array indicating the parameters of the keys used with the ECDH algorithm in the OSCORE group identified by 'gname'. Its content depends on the value of 'ecdh_alg'. In particular, its format and value are the same of the COSE capabilities array for the COSE key type of the keys used with the algorithm indicated in 'ecdh_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types].
- * The fifth element 'cred_fmt' is a CBOR integer indicating the format of authentication credentials used in the OSCORE group identified by 'gname'. It takes value from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters] (REQ6). Acceptable values denote a format that MUST provide the

public key as well as the comprehensive set of information related to the public key algorithm, including, e.g., the used elliptic curve (when applicable). The same considerations and guidelines for the 'pub_key_enc' element of 'sign_info' apply (see Section 5.3).

The CDDL notation [RFC8610] of the 'ecdh_info' parameter is given below.

```
ecdh_info = ecdh_info_req / ecdh_info_resp

ecdh_info_req = null                                ; in the Token Transfer
                                                       ; Request to the
                                                       ; Group Manager

ecdh_info_res = [ + ecdh_info_entry ] ; in the Token Transfer
                                           ; Response from the
                                           ; Group Manager

ecdh_info_entry =
[
  id : gname / [ + gname ],
  ecdh_alg : int / tstr,
  ecdh_parameters : [ any ],
  ecdh_key_parameters : [ any ],
  cred_fmt = int
]

gname = tstr
```

This format is consistent with every ECDH algorithm currently defined in [I-D.ietf-cose-rfc8152bis-algs], i.e., with algorithms that have only the COSE key type as their COSE capability. Appendix B of this document describes how the format of each 'ecdh_info_entry' can be generalized for possible future registered algorithms having a different set of COSE capabilities.

5.3.2. 'kdc_dh_creds' Parameter

The 'kdc_dh_creds' parameter is an OPTIONAL parameter of the request and response messages exchanged between the Client and the authz-info endpoint at the RS (see Section 5.10.1. of [I-D.ietf-ace-oauth-authz]).

This parameter allows the Client to request and retrieve the Diffie-Hellman authentication credentials of the RS, i.e., authentication credentials including a Diffie-Hellman public key of the RS.

In this application profile, this parameter is used to request and retrieve from the Group Manager its Diffie-Hellman authentication credentials to use, in the OSCORE groups that the Client has been authorized to join. The Group Manager has specifically a Diffie-Hellman authentication credential in an OSCORE group, and thus a Diffie-Hellman public key in that group, if and only if the group is a pairwise-only group. In this case, the early retrieval of the Group Manager's authentication credential is necessary in order for the joining node to prove the possession of its own private key, upon joining the group (see Section 6.1).

When used in the Token Transfer Request sent to the Group Manager, the 'kdc_dh_creds' parameter has value the CBOR simple value "null" (0xf6). This is done to ask for the Diffie-Hellman authentication credentials that the Group Manager uses in the OSCORE groups that the Client has been authorized to join.

When used in the following Token Transfer Response from the Group Manager, the 'kdc_dh_creds' parameter is a CBOR array of one or more elements. The number of elements is at most the number of OSCORE groups that the Client has been authorized to join.

Each element 'kdc_dh_creds_entry' contains information about the Group Manager's Diffie-Hellman authentication credentials, for one or more OSCORE groups that are pairwise-only groups and that the Client has been authorized to join. Each element is formatted as follows.

- * The first element 'id' is the group name of the OSCORE group or an array of group names for the OSCORE groups for which the specified information applies. In particular 'id' MUST refer exclusively to OSCORE groups that are pairwise-only groups.
- * The second element 'cred_fmt' is a CBOR integer indicating the format of authentication credentials used in the OSCORE group identified by 'gname'. It takes value from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters] (REQ6). Acceptable values denote a format that MUST explicitly provide the public key as well as comprehensive set of information related to the public key algorithm, including, e.g., the used elliptic curve (when applicable). The same considerations and guidelines for the 'pub_key_enc' element of 'sign_info' apply (see Section 5.3).
- * The third element 'cred' is a CBOR byte string, which encodes the Group Manager's Diffie-Hellman authentication credential in its original binary representation made available to other endpoints in the group. In particular, the original binary representation complies with the format specified by the 'cred_fmt' element.

Note that the authentication credential provides the comprehensive set of information related to its public key algorithm, i.e., the ECDH algorithm used in the OSCORE group as pairwise key agreement algorithm.

The CDDL notation [RFC8610] of the 'kdc_dh_creds' parameter is given below.

```
kdc_dh_creds = kdc_dh_creds_req / kdc_dh_creds_resp
```

```
kdc_dh_creds_req = null                                ; in the Token Transfer
                                                         ; Request to the
                                                         ; Group Manager
```

```
kdc_dh_creds_res = [ + kdc_dh_creds_entry ] ; in the Token Transfer
                                                         ; Response from the
                                                         ; Group Manager
```

```
kdc_dh_creds_entry =
[
  id : gname / [ + gname ],
  cred_fmt = int,
  cred = bstr
]
```

```
gname = tstr
```

6. Group Joining

This section describes the interactions between the joining node and the Group Manager to join an OSCORE group. The message exchange between the joining node and the Group Manager consists of the messages defined in Section 4.3.1.1 of [I-D.ietf-ace-key-groupcomm]. Note that what is defined in [I-D.ietf-ace-key-groupcomm] applies, and only additions or modifications to that specification are defined in this document.

6.1. Send the Joining Request

The joining node requests to join the OSCORE group by sending a Joining Request message to the related group-membership resource at the Group Manager, as per Section 4.3.1.1 of [I-D.ietf-ace-key-groupcomm]. Additionally to what is defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], the following applies.

- * The 'scope' parameter MUST be included. Its value encodes one scope entry with the format defined in Section 3, indicating the group name and the role(s) that the joining node wants to take in the group.
- * The 'get_pub_keys' parameter is present only if the joining node wants to retrieve the authentication credentials of the group members from the Group Manager during the joining process (see Section 4). Otherwise, this parameter MUST NOT be present.

If this parameter is present and its value is not the CBOR simple value "null" (0xf6), each element of the inner CBOR array 'role_filter' is encoded as a CBOR unsigned integer, with the same value of a permission set ("Tperm") indicating that role or combination of roles in a scope entry, as defined in Section 3.

- * 'cnonce' contains a dedicated nonce N_C generated by the joining node. For the N_C value, it is RECOMMENDED to use a 8-byte long random nonce.
- * The proof-of-possession (PoP) evidence included in 'client_cred_verify' is computed as defined below (REQ14). In either case, the N_S used to build the PoP input is as defined in Section 6.1.1.
 - If the group is not a pairwise-only group, the PoP evidence MUST be a signature. The joining node computes the signature by using the same private key and signature algorithm it intends to use for signing messages in the OSCORE group.
 - If the group is a pairwise-only group, the PoP evidence MUST be a MAC computed as follows, by using the HKDF Algorithm HKDF SHA-256, which consists of composing the HKDF-Extract and HKDF-Expand steps [RFC5869].

MAC = HKDF(salt, IKM, info, L)

The input parameters of HKDF are as follows.

- o salt takes as value the empty byte string.
- o IKM is computed as a cofactor Diffie-Hellman shared secret, see Section 5.7.1.2 of [NIST-800-56A], using the ECDH algorithm used in the OSCORE group. The joining node uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the Group Manager. For X25519 and X448, the procedure is described in Section 5 of [RFC7748].

- o info takes as value the PoP input.
- o L is equal to 8, i.e., the size of the MAC, in bytes.

6.1.1. Value of the N_S Challenge

The value of the N_S challenge is determined as follows.

1. If the joining node has provided the Access Token to the Group Manager by means of a Token Transfer Request to the /authz-info endpoint as in Section 5.3, then N_S takes the same value of the most recent 'kdcchallenge' parameter received by the joining node from the Group Manager. This can be either the one specified in the Token Transfer Response, or the one possibly specified in a 4.00 (Bad Request) error response to a following Joining Request (see Section 6.2).
2. If the provisioning of the Access Token to the Group Manager has relied on the DTLS profile of ACE [I-D.ietf-ace-dtls-authorize] with the Access Token as content of the "psk_identity" field of the ClientKeyExchange message [RFC6347], then N_S is an exporter value computed as defined in Section 7.5 of [RFC8446]. Specifically, N_S is exported from the DTLS session between the joining node and the Group Manager, using an empty 'context_value', 32 bytes as 'key_length', and the exporter label "EXPORTER-ACE-Sign-Challenge-coap-group-oscore-app" defined in Section 16.7 of this document.

It is up to applications to define how N_S is computed in further alternative settings.

Section 15.3 provides security considerations on the reuse of the N_S challenge.

6.2. Receive the Joining Request

The Group Manager processes the Joining Request as defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], with the following additions.

The Group Manager verifies the PoP evidence contained in 'client_cred_verify' as follows:

- * As PoP input, the Group Manager uses the value of the 'scope' parameter from the Joining Request as a CBOR byte string, concatenated with N_S encoded as a CBOR byte string, concatenated with N_C encoded as a CBOR byte string. In particular, N_S is determined as described in Section 6.1.1, while N_C is the nonce provided in the 'cnonce' parameter of the Joining Request.
- * As public key of the joining node, the Group Manager uses either the one included in the authentication credential retrieved from the 'client_cred' parameter of the Joining Request, or the one from the already stored authentication credential as acquired from previous interactions with the joining node (see Section 4).
- * If the group is not a pairwise-only group, the PoP evidence is a signature. The Group Manager verifies it by using the public key of the joining node, as well as the signature algorithm used in the OSCORE group and possible corresponding parameters.
- * If the group is a pairwise-only group, the PoP evidence is a MAC. The Group Manager recomputes the MAC through the same process taken by the joining node when preparing the value of the 'client_cred_verify' parameter for the Joining Request (see Section 6.1), with the difference that the Group Manager uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the joining node. The verification succeeds if and only if the recomputed MAC is equal to the MAC conveyed as PoP evidence in the Joining Request.

The Group Manager MUST reply with a 5.03 (Service Unavailable) error response in the following cases:

- * There are currently no OSCORE Sender IDs available to assign in the OSCORE group and, at the same time, the joining node is not going to join the group exclusively as monitor. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 4 ("No available node identifiers").
- * The OSCORE group that the joining node has been trying to join is currently inactive (see Section 8.1). The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 9 ("Group currently not active").

The Group Manager MUST reply with a 4.00 (Bad Request) error response in the following cases:

- * The 'client_cred' parameter is present in the Joining Request and its value is not an eligible authentication credential (e.g., it is not of the format accepted in the group).
- * The 'client_cred' parameter is not present in the Joining Request while the joining node is not going to join the group exclusively as monitor, and any of the following conditions holds:
 - The Group Manager does not store an eligible authentication credential (e.g., of the format accepted in the group) for the joining node.
 - The Group Manager stores multiple eligible authentication credentials (e.g., of the format accepted in the group) for the joining node.
- * The 'scope' parameter is not present in the Joining Request, or it is present and specifies any set of roles not included in the following list: "requester", "responder", "monitor", ("requester", "responder"). Future specifications that define a new role for members of OSCORE groups MUST define possible sets of roles (including the new role and existing roles) that are acceptable to specify in the 'scope' parameter of a Joining Request.
- * The Joining Request includes the 'client_cred' parameter but does not include both the 'cnonce' and 'client_cred_verify' parameters.

In order to prevent the acceptance of Ed25519 and Ed448 public keys that cannot be successfully converted to Montgomery coordinates, and thus cannot be used for the derivation of pairwise keys (see Section 2.4.1 of [I-D.ietf-core-oscore-groupcomm]), the Group Manager MAY reply with a 4.00 (Bad Request) error response in case all the following conditions hold:

- * The OSCORE group uses the pairwise mode of Group OSCORE.
- * The OSCORE group uses EdDSA public keys [RFC8032].
- * The authentication credential of the joining node from the 'client_cred' parameter includes a public key which:
 - Is for the elliptic curve Ed25519 and has its Y coordinate equal to -1 or $1 \pmod{p}$, with $p = (2^{255} - 19)$, see Section 4.1 of [RFC7748]; or
 - Is for the elliptic curve Ed448 and has its Y coordinate equal to -1 or $1 \pmod{p}$, with $p = (2^{448} - 2^{224} - 1)$, see Section 4.2 of [RFC7748].

A 4.00 (Bad Request) error response from the Group Manager to the joining node MUST have content format `application/ace-groupcomm+cbor`. The response payload is a CBOR map formatted as follows:

- * If the group uses (also) the group mode of Group OSCORE, the CBOR map MUST contain the `'sign_info'` parameter, whose CBOR label is defined in Section 8 of [I-D.ietf-ace-key-groupcomm]. This parameter has the same format of `'sign_info_res'` defined in Section 3.3.1 of [I-D.ietf-ace-key-groupcomm]. In particular, it includes a single element `'sign_info_entry'` pertaining to the OSCORE group that the joining node has tried to join with the Joining Request.
- * If the group uses (also) the pairwise mode of Group OSCORE, the CBOR map MUST contain the `'ecdh_info'` parameter, whose CBOR label is defined in Section 16.3. This parameter has the same format of `'ecdh_info_res'` defined in Section 5.3.1. In particular, it includes a single element `'ecdh_info_entry'` pertaining to the OSCORE group that the joining node has tried to join with the Joining Request.
- * If the group is a pairwise-only group, the CBOR map MUST contain the `'kdc_dh_creds'` parameter, whose CBOR label is defined in Section 16.3. This parameter has the same format of `'kdc_dh_creds_res'` defined in Section 5.3.2. In particular, it includes a single element `'kdc_dh_creds_entry'` pertaining to the OSCORE group that the joining node has tried to join with the Joining Request.
- * The CBOR map MAY include the `'kdcchallenge'` parameter, whose CBOR label is defined in Section 8 of [I-D.ietf-ace-key-groupcomm]. If present, this parameter is a CBOR byte string, which encodes a newly generated `'kdcchallenge'` value that the Client can use when preparing a Joining Request (see Section 6.1). In such a case the Group Manager MUST store the newly generated value as the `'kdcchallenge'` value associated with the joining node, possibly replacing the currently stored value.

6.2.1. Follow-up to a 4.00 (Bad Request) Error Response

When receiving a 4.00 (Bad Request) error response, the joining node MAY send a new Joining Request to the Group Manager. In such a case:

- * The `'nonce'` parameter MUST include a new dedicated nonce `N_C` generated by the joining node.

- * The 'client_cred' parameter MUST include an authentication credential in the format indicated by the Group Manager. Also, the authentication credential as well as the included public key MUST be compatible with the signature or ECDH algorithm, and possible associated parameters.
- * The 'client_cred_verify' parameter MUST include a PoP evidence computed as described in Section 6.1, by using the private key associated with the authentication credential specified in the current 'client_cred' parameter, with the signature or ECDH algorithm, and possible associated parameters indicated by the Group Manager. If the error response from the Group Manager includes the 'kdcchallenge' parameter, the joining node MUST use its content as new N_S challenge to compute the PoP evidence.

6.3. Send the Joining Response

If the processing of the Joining Request described in Section 6.2 is successful, the Group Manager updates the group membership by registering the joining node NODENAME as a new member of the OSCORE group GROUPNAME, as described in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm].

If the joining node has not taken exclusively the role of monitor, the Group Manager performs also the following actions.

- * The Group Manager selects an available OSCORE Sender ID in the OSCORE group, and exclusively assigns it to the joining node. The Group Manager MUST NOT assign an OSCORE Sender ID to the joining node if this joins the group exclusively with the role of monitor, according to what is specified in the Access Token (see Section 5.2).

Consistently with Section 3.2.1 of [I-D.ietf-core-oscore-groupcomm], the Group Manager MUST assign an OSCORE Sender ID that has not been used in the OSCORE group since the latest time when the current Gid value was assigned to the group.

If the joining node is recognized as a current group member, e.g., through the ongoing secure communication association, the following also applies.

- The Group Manager MUST assign a new OSCORE Sender ID different than the one currently used by the joining node in the OSCORE group.

- The Group Manager MUST add the old, relinquished OSCORE Sender ID of the joining node to the most recent set of stale Sender IDs, in the collection associated with the group (see Section 7.1).
- * The Group Manager stores the association between i) the authentication credential of the joining node; and ii) the Group Identifier (Gid), i.e., the OSCORE ID Context, associated with the OSCORE group together with the OSCORE Sender ID assigned to the joining node in the group. The Group Manager MUST keep this association updated over time.

Then, the Group Manager replies to the joining node, providing the updated security parameters and keying material necessary to participate in the group communication. This success Joining Response is formatted as defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], with the following additions:

- * The 'gkty' parameter identifies a key of type "Group_OSCORE_Input_Material object", defined in Section 16.4 of this document.
- * The 'key' parameter includes what the joining node needs in order to set up the Group OSCORE Security Context as per Section 2 of [I-D.ietf-core-oscore-groupcomm].

This parameter has as value a Group_OSCORE_Input_Material object, which is defined in this document and extends the OSCORE_Input_Material object encoded in CBOR as defined in Section 3.2.1 of [I-D.ietf-ace-oscore-profile]. In particular, it contains the additional parameters 'group_senderId', 'cred_fmt', 'sign_enc_alg', 'sign_alg', 'sign_params', 'ecdh_alg' and 'ecdh_params' defined in Section 16.6 of this document.

More specifically, the 'key' parameter is composed as follows.

- The 'hkdf' parameter, if present, specifies the HKDF Algorithm used in the OSCORE group. The HKDF Algorithm is specified by the HMAC Algorithm value. This parameter MAY be omitted, if the HKDF Algorithm used in the group is HKDF SHA-256. Otherwise, this parameter MUST be present.
- The 'salt' parameter, if present, has as value the OSCORE Master Salt used in the OSCORE group. This parameter MAY be omitted, if the Master Salt used in the group is the empty byte string. Otherwise, this parameter MUST be present.

- The 'ms' parameter includes the OSCORE Master Secret value used in the OSCORE group. This parameter MUST be present.
- The 'contextId' parameter has as value the Group Identifier (Gid), i.e., the OSCORE ID Context of the OSCORE group. This parameter MUST be present.
- The 'group_senderId' parameter has as value the OSCORE Sender ID assigned to the joining node by the Group Manager, as described above. This parameter MUST be present if and only if the node does not join the OSCORE group exclusively with the role of monitor, according to what is specified in the Access Token (see Section 5.2).
- The 'cred_fmt' parameter specifies the format of authentication credentials used in the OSCORE group. This parameter MUST be present and it takes value from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters] (REQ6). Consistently with Section 2.3 of [I-D.ietf-core-oscore-groupcomm], acceptable values denote a format that MUST explicitly provide the public key as well as the comprehensive set of information related to the public key algorithm, including, e.g., the used elliptic curve (when applicable).

At the time of writing this specification, acceptable formats of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC7925] and C509 certificates [I-D.ietf-cose-cbor-encoded-cert]. Further formats may be available in the future, and would be acceptable to use as long as they comply with the criteria defined above.

[As to CWTs and CCSs, the COSE Header Parameters 'kcwt' and 'kccs' are under pending registration requested by draft-ietf-lake-edhoc.]

[As to C509 certificates, the COSE Header Parameters 'c5b' and 'c5c' are under pending registration requested by draft-ietf-cose-cbor-encoded-cert.]

The 'key' parameter MUST also include the following parameters, if and only if the OSCORE group is not a pairwise-only group.

- The 'sign_enc_alg' parameter, specifying the Signature Encryption Algorithm used in the OSCORE group to encrypt messages protected with the group mode. This parameter takes values from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].
- The 'sign_alg' parameter, specifying the Signature Algorithm used to sign messages in the OSCORE group. This parameter takes values from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].
- The 'sign_params' parameter, specifying the parameters of the Signature Algorithm. This parameter is a CBOR array, which includes the following two elements:
 - o 'sign_alg_capab': a CBOR array, with the same format and value of the COSE capabilities array for the Signature Algorithm indicated in 'sign_alg', as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms].
 - o 'sign_key_type_capab': a CBOR array, with the same format and value of the COSE capabilities array for the COSE key type of the keys used with the Signature Algorithm indicated in 'sign_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types].

The 'key' parameter MUST also include the following parameters, if and only if the OSCORE group is not a signature-only group.

- The 'alg' parameter, specifying the AEAD Algorithm used in the OSCORE group to encrypt messages protected with the pairwise mode.
- The 'ecdh_alg' parameter, specifying the Pairwise Key Agreement Algorithm used in the OSCORE group. This parameter takes values from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].
- The 'ecdh_params' parameter, specifying the parameters of the Pairwise Key Agreement Algorithm. This parameter is a CBOR array, which includes the following two elements:

- o `'ecdh_alg_capab'`: a CBOR array, with the same format and value of the COSE capabilities array for the algorithm indicated in `'ecdh_alg'`, as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms].
- o `'ecdh_key_type_capab'`: a CBOR array, with the same format and value of the COSE capabilities array for the COSE key type of the keys used with the algorithm indicated in `'ecdh_alg'`, as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types].

The format of `'key'` defined above is consistent with every signature algorithm and ECDH algorithm currently considered in [I-D.ietf-cose-rfc8152bis-algs], i.e., with algorithms that have only the COSE key type as their COSE capability. Appendix B of this document describes how the format of the `'key'` parameter can be generalized for possible future registered algorithms having a different set of COSE capabilities.

Furthermore, the following applies.

- * The `'exp'` parameter MUST be present.
- * The `'ace-groupcomm-profile'` parameter MUST be present and has value `coap_group_oscore_app` (PROFILE_TBD), which is defined in Section 16.5 of this document.
- * The `'pub_keys'` parameter, if present, includes the authentication credentials requested by the joining node by means of the `'get_pub_keys'` parameter in the Joining Request.

If the joining node has asked for the authentication credentials of all the group members, i.e., `'get_pub_keys'` had value the CBOR simple value `"null"` (0xf6) in the Joining Request, then the Group Manager provides only the authentication credentials of the group members that are relevant to the joining node. That is, in such a case, `'pub_keys'` includes only: i) the authentication credentials of the responders currently in the OSCORE group, in case the joining node is configured (also) as requester; and ii) the authentication credentials of the requesters currently in the OSCORE group, in case the joining node is configured (also) as responder or monitor.

- * The 'peer_identifiers' parameter includes the OSCORE Sender ID of each group member whose authentication credential is specified in the 'pub_keys' parameter. That is, a group member's Sender ID is used as identifier for that group member (REQ25).
- * The 'group_policies' parameter SHOULD be present, and SHOULD include the following elements:
 - "Key Update Check Interval" defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], with default value 3600;
 - "Expiration Delta" defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], with default value 0.
- * The 'kdc_cred' parameter MUST be present, specifying the Group Manager's authentication credential in its original binary representation (REQ8). The Group Manager's authentication credential MUST be in the format used in the OSCORE group. Also, the authentication credential as well as the included public key MUST be compatible with the signature or ECDH algorithm, and possible associated parameters used in the OSCORE group.
- * The 'kdc_nonce' parameter MUST be present, specifying the dedicated nonce N_KDC generated by the Group Manager. For N_KDC, it is RECOMMENDED to use a 8-byte long random nonce.
- * The 'kdc_cred_verify' parameter MUST be present, specifying the proof-of-possession (PoP) evidence computed by the Group Manager. The PoP evidence is computed over the nonce N_KDC, which is specified in the 'kdc_nonce' parameter and taken as PoP input. The PoP evidence is computed as defined below (REQ21).
 - If the group is not a pairwise-only group, the PoP evidence MUST be a signature. The Group Manager computes the signature by using the signature algorithm used in the OSCORE group, as well as its own private key associated with the authentication credential specified in the 'kdc_cred' parameter.
 - If the group is a pairwise-only group, the PoP evidence MUST be a MAC computed as follows, by using the HKDF Algorithm HKDF SHA-256, which consists of composing the HKDF-Extract and HKDF-Expand steps [RFC5869].

MAC = HKDF(salt, IKM, info, L)

The input parameters of HKDF are as follows.
 - o salt takes as value the empty byte string.

- o IKM is computed as a cofactor Diffie-Hellman shared secret, see Section 5.7.1.2 of [NIST-800-56A], using the ECDH algorithm used in the OSCORE group. The Group Manager uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the joining node. For X25519 and X448, the procedure is described in Section 5 of [RFC7748].
 - o info takes as value the PoP input.
 - o L is equal to 8, i.e., the size of the MAC, in bytes.
- * The 'group_rekeying' parameter MAY be omitted, if the Group Manager uses the "Point-to-Point" group rekeying scheme registered in Section 11.14 of [I-D.ietf-ace-key-groupcomm] as rekeying scheme in the OSCORE group (OPT9). Its detailed use for this profile is defined in Section 11 of this document. In any other case, the 'group_rekeying' parameter MUST be included.

As a last action, if the Group Manager reassigns Gid values during the group's lifetime (see Section 3.2.1.1 of [I-D.ietf-core-oscore-groupcomm]), then the Group Manager MUST store the Gid specified in the 'contextId' parameter of the 'key' parameter, as the Birth Gid of the joining node in the joined group (see Section 3 of [I-D.ietf-core-oscore-groupcomm]). This applies also in case the joining node is in fact re-joining the group; in such a case, the newly determined Birth Gid overwrites the one currently stored.

6.4. Receive the Joining Response

Upon receiving the Joining Response, the joining node retrieves the Group Manager's authentication credential from the 'kdc_cred' parameter. The joining node MUST verify the proof-of-possession (PoP) evidence specified in the 'kdc_cred_verify' parameter of the Joining Response as defined below (REQ21).

- * If the group is not a pairwise-only group, the PoP evidence is a signature. The joining node verifies it by using the public key of the Group Manager from the received authentication credential, as well as the signature algorithm used in the OSCORE group and possible corresponding parameters.
- * If the group is a pairwise-only group, the PoP evidence is a MAC. The joining node recomputes the MAC through the same process taken by the Group Manager when computing the value of the 'kdc_cred_verify' parameter (see Section 6.3), with the difference that the joining node uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the Group Manager from the

received authentication credential. The verification succeeds if and only if the recomputed MAC is equal to the MAC conveyed as PoP evidence in the Joining Response.

In case of failed verification of the PoP evidence, the joining node MUST stop processing the Joining Response and MAY send a new Joining Request to the Group Manager (see Section 6.1).

In case of successful verification of the PoP evidence, the joining node uses the information received in the Joining Response to set up the Group OSCORE Security Context, as described in Section 2 of [I-D.ietf-core-oscore-groupcomm]. If the following parameters were not included in the 'key' parameter of the Joining Response, the joining node considers the default values specified below, consistently with Section 3.2 of [RFC8613].

- * Absent the 'hkdf' parameter, the joining node considers HKDF SHA-256 as HKDF Algorithm to use in the OSCORE group.
- * Absent the 'salt' parameter, the joining node considers the empty byte string as Master Salt to use in the OSCORE group.
- * Absent the 'group_rekeying' parameter, the joining node considers the "Point-to-Point" group rekeying scheme registered in Section 11.14 of [I-D.ietf-ace-key-groupcomm] as the rekeying scheme used in the group (OPT9). Its detailed use for this profile is defined in Section 11 of this document.

In addition, the joining node maintains an association between each authentication credential retrieved from the 'pub_keys' parameter and the role(s) that the corresponding group member has in the OSCORE group.

From then on, the joining node can exchange group messages secured with Group OSCORE as described in [I-D.ietf-core-oscore-groupcomm]. When doing so:

- * The joining node MUST NOT process an incoming request message, if protected by a group member whose authentication credential is not associated with the role "Requester".
- * The joining node MUST NOT process an incoming response message, if protected by a group member whose authentication credential is not associated with the role "Responder".
- * The joining node MUST NOT use the pairwise mode of Group OSCORE to process messages in the group, if the Joining Response did not include the 'ecdh_alg' parameter.

If the application requires backward security, the Group Manager MUST generate updated security parameters and group keying material, and provide it to the current group members, upon the new node's joining (see Section 11). In such a case, the joining node is not able to access secure communication in the OSCORE group occurred prior its joining.

7. Overview of the Group Rekeying Process

In a number of cases, the Group Manager has to generate new keying material and distribute it to the group (rekeying), as also discussed in Section 3.2 of [I-D.ietf-core-oscore-groupcomm].

To this end the Group Manager MUST support the Group Rekeying Process described in Section 11 of this document, as an instance of the "Point-to-Point" rekeying scheme defined in Section 6.1 of [I-D.ietf-ace-key-groupcomm] and registered in Section 11.14 of [I-D.ietf-ace-key-groupcomm]. Future documents may define the use of alternative group rekeying schemes for this application profile, together with the corresponding rekeying message formats. The resulting group rekeying process MUST comply with the functional steps defined in Section 3.2 of [I-D.ietf-core-oscore-groupcomm].

Upon generating the new group keying material and before starting its distribution, the Group Manager MUST increment the version number of the group keying material. When rekeying a group, the Group Manager MUST preserve the current value of the OSCORE Sender ID of each member in that group.

The data distributed to a group through a rekeying MUST include:

- * The new version number of the group keying material for the group.
- * A new Group Identifier (Gid) for the group as introduced in [I-D.ietf-ace-key-groupcomm], used as ID Context parameter of the Group OSCORE Common Security Context of that group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]).

Note that the Gid differs from the group name also introduced in [I-D.ietf-ace-key-groupcomm], which is a plain, stable and invariant identifier, with no cryptographic relevance and meaning.

- * A new value for the Master Secret parameter of the Group OSCORE Common Security Context of the group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]).

- * A set of stale Sender IDs, which allows each rekeyed node to purge authentication credentials and Recipient Contexts used in the group and associated with those Sender IDs. This in turn allows every group member to rely on stored authentication credentials, in order to confidently assert the group membership of other sender nodes, when receiving protected messages in the group (see Section 3.2 of [I-D.ietf-core-oscore-groupcomm]). More details on the maintenance of stale Sender IDs are provided in Section 7.1.

Also, the data distributed through a group rekeying MAY include a new value for the Master Salt parameter of the Group OSCORE Common Security Context of that group.

The Group Manager MUST rekey the group in the following cases.

- * The application requires backward security - In this case, the group is rekeyed when a node joins the group as a new member. Therefore, a joining node cannot access communications in the group prior its joining.
- * One or more nodes leave the group - That is, the group is rekeyed when one or more current members spontaneously request to leave the group (see Section 9.11), or when the Group Manager forcibly evicts them from the group, e.g., due to expired or revoked authorization (see Section 10). Therefore, a leaving node cannot access communications in the group after its leaving, thus ensuring forward security in the group.

Due to the set of stale Sender IDs distributed through the rekeying, this ensures that a node owning the latest group keying material does not store the authentication credentials of former group members (see Sections 3.2 and 12.1 of [I-D.ietf-core-oscore-groupcomm]).

- * Extension of group lifetime - That is, the group is rekeyed when the expiration time for the group keying material approaches or has passed, if it is appropriate to extend the group operation beyond that.

The Group Manager MAY rekey the group for other reasons, e.g., according to an application-specific rekeying period or scheduling.

7.1. Stale OSCORE Sender IDs

Throughout the lifetime of every group, the Group Manager MUST maintain a collection of stale Sender IDs for that group.

The collection associated with a group MUST include up to $N > 1$ ordered sets of stale OSCORE Sender IDs. It is up to the application to specify the value of N , possibly on a per-group basis.

The N -th set includes the Sender IDs that have become "stale" under the current version V of the group keying material. The $(N - 1)$ -th set refers to the immediately previous version $(V - 1)$ of the group keying material, and so on.

In the following cases, the Group Manager MUST add a new element to the most recent set X , i.e., the set associated with the current version V of the group keying material.

- * When a current group member obtains a new Sender ID, its old Sender ID is added to X . This happens when the Group Manager assigns a new Sender ID upon request from the group member (see Section 9.2), or in case the group member re-joins the group (see Section 6.1 and Section 6.3), thus also obtaining a new Sender ID.
- * When a current group member leaves the group, its current Sender ID is added to X . This happens when a group member requests to leave the group (see Section 9.11) or is forcibly evicted from the group (see Section 10).

The value of N can change throughout the lifetime of the group. If the new value N' is smaller than N , the Group Manager MUST preserve the (up to) N' most recent sets in the collection and MUST delete any possible older set from the collection.

Finally, the Group Manager MUST perform the following actions, when the group is rekeyed and the group shifts to the next version $V' = (V + 1)$ of the group keying material.

1. The Group Manager rekeys the group. This includes also distributing the set of stale Sender IDs X associated with the old group keying material with version V (see Section 7).
2. After completing the group rekeying, the Group Manager creates a new empty set X' associated with the new version V' of the newly established group keying material, i.e., $V' = (V + 1)$.
3. If the current collection of stale Sender IDs has size N , the Group Manager deletes the oldest set in the collection.
4. The Group Manager adds the new set X' to the collection of stale Sender IDs, as the most recent set.

8. Interface at the Group Manager

The Group Manager provides the interface defined in Section 4.1 of [I-D.ietf-ace-key-groupcomm], with the additional sub-resources defined from Section 8.1 to Section 8.3 of this document.

Furthermore, Section 8.4 provides a summary of the CoAP methods admitted to access different resources at the Group Manager, for nodes with different roles in the group or as non members (REQ11).

The GROUPNAME segment of the URI path MUST match with the group name specified in the scope entry of the Access Token scope (i.e., 'gname' in Section 3.1 of [I-D.ietf-ace-key-groupcomm]) (REQ7).

The Resource Type (rt=) Link Target Attribute value "core.osc.gm" is registered in Section 16.11 (REQ10), and can be used to describe group-membership resources and its sub-resources at a Group Manager, e.g., by using a link-format document [RFC6690].

Applications can use this common resource type to discover links to group-membership resources for joining OSCORE groups, e.g., by using the approach described in [I-D.tiloca-core-oscore-discovery].

8.1. ace-group/GROUPNAME/active

This resource implements a GET handler.

8.1.1. GET Handler

The handler expects a GET request.

In addition to what is defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm], the handler verifies that the requesting Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler replies with a 2.05 (Content) response, specifying the current status of the group, i.e., active or inactive. The payload of the response is formatted as defined in Section 9.9.

The method to set the current group status is out of the scope of this document, and is defined for the administrator interface of the Group Manager specified in [I-D.ietf-ace-oscore-gm-admin].

8.2. ace-group/GROUPNAME/verif-data

This resource implements a GET handler.

8.2.1. GET Handler

The handler expects a GET request.

In addition to what is defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm], the Group Manager performs the following checks.

If the requesting Client is a current group member, the Group Manager MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 8 ("Operation permitted only to signature verifiers").

If GROUPNAME denotes a pairwise-only group, the Group Manager MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 7 ("Signatures not used in the group").

If all verifications succeed, the handler replies with a 2.05 (Content) response, specifying data that allow also an external signature verifier to verify signatures of messages protected with the group mode and sent to the group (see Sections 3.1 and 8.5 of [I-D.ietf-core-oscore-groupcomm]). The response MUST have Content-Format set to application/ace-groupcomm+cbor. The payload of the response is a CBOR map, which is formatted as defined in Section 9.6.

8.3. ace-group/GROUPNAME/stale-sids

This resource implements a FETCH handler.

8.3.1. FETCH Handler

The handler expects a FETCH request, whose payload specifies a version number of the group keying material, encoded as an unsigned CBOR integer.

In addition to what is defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm], the handler verifies that the requesting Client is a current member of the group. If the verification fails, the Group Manager MUST reply with a 4.03

(Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler replies with a 2.05 (Content) response, specifying data that allow the requesting Client to delete the Recipient Contexts and authentication credentials associated with former members of the group (see Section 3.2 of [I-D.ietf-core-oscore-groupcomm]). The payload of the response is formatted as defined in Section 11.3.1.

8.4. Admitted Methods

The table in Figure 2 summarizes the CoAP methods admitted to access different resources at the Group Manager, for (non-)members of a group with group name GROUPNAME, and considering different roles. The last two rows of the table apply to a node with node name NODENAME.

Resource	Type1	Type2	Type3	Type4
ace-group/	F	F	F	F
ace-group/GROUPNAME/	G Po	G Po	Po *	Po
ace-group/GROUPNAME/active	G	G	-	-
ace-group/GROUPNAME/verif-data	-	-	G	-
ace-group/GROUPNAME/pub-key	G F	G F	G F	-
ace-group/GROUPNAME/kdc-pub-key	G	G	G	-
ace-group/GROUPNAME/stale-sids	F	F	-	-
ace-group/GROUPNAME/policies	G	G	-	-
ace-group/GROUPNAME/num	G	G	-	-
ace-group/GROUPNAME/nodes/ NODENAME	G Pu D	G D	-	-
ace-group/GROUPNAME/nodes/ NODENAME/pub-key	Po	-	-	-

CoAP methods: G = GET; F = FETCH; Po = POST; Pu = PUT; D = DELETE

Type1 = Member as Requester and/or Responder

Type2 = Member as Monitor

Type3 = Non-member (authorized to be signature verifier)

(*) = cannot join the group as signature verifier

Type4 = Non-member (not authorized to be signature verifier)

Figure 2: Admitted CoAP Methods on the Group Manager Resources

8.4.1. Signature Verifiers

Just like any candidate group member, a signature verifier provides the Group Manager with an Access Token, as described in Section 5.3. However, unlike candidate group members, it does not join any OSCORE group, i.e., it does not perform the joining process defined in Section 6.

After successfully transferring an Access Token to the Group Manager, a signature verifier is allowed to perform only some operations as non-member of a group, and only for the OSCORE groups specified in the validated Access Token. These are the operations specified in Section 9.3, Section 9.5, Section 9.6 and Section 9.10.

Consistently, in case a node is not a member of the group with group name GROUPNAME and is authorized to be only signature verifier for that group, the Group Manager MUST reply with a 4.03 (Forbidden) error response if that node attempts to access any other endpoint than: /ace-group; ace-group/GROUPNAME/verif-data; /ace-group/GROUPNAME/pub-key; and ace-group/GROUPNAME/kdc-pub-key.

8.5. Operations Supported by Clients

Building on what is defined in Section 4.1.1 of [I-D.ietf-ace-key-groupcomm], and with reference to the resources at the Group Manager newly defined earlier in Section 8 of this document, it is expected that a Client minimally supports also the following set of operations and corresponding interactions with the Group Manager (REQ12).

- * GET request to ace-group/GROUPNAME/active, in order to check the current status of the group.
- * GET request to ace-group/GROUPNAME/verif-data, in order for a signature verifier to retrieve data required to verify signatures of messages protected with the group mode of Group OSCORE and sent to a group (see Sections 3.1 and 8.5 of [I-D.ietf-core-oscore-groupcomm]). Note that this operation is relevant to support only to signature verifiers.
- * FETCH request to ace-group/GROUPNAME/stale-sids, in order to retrieve from the Group Manager the data required to delete some of the stored group members' authentication credentials and associated Recipient Contexts (see Section 8.3.1). These data are provided as an aggregated set of stale Sender IDs, which are used as specified in Section 11.3.

9. Additional Interactions with the Group Manager

This section defines the possible interactions with the Group Manager, in addition to the group joining specified in Section 6.

9.1. Retrieve Updated Keying Material

At some point, a group member considers the Group OSCORE Security Context invalid and to be renewed. This happens, for instance, after a number of unsuccessful security processing of incoming messages from other group members, or when the Security Context expires as specified by the 'exp' parameter of the Joining Response.

When this happens, the group member retrieves updated security parameters and group keying material. This can occur in the two different ways described below.

9.1.1. Get Group Keying Material

If the group member wants to retrieve only the latest group keying material, it sends a Key Distribution Request to the Group Manager.

In particular, it sends a CoAP GET request to the endpoint /ace-group/GROUPNAME at the Group Manager.

The Group Manager processes the Key Distribution Request according to Section 4.3.2 of [I-D.ietf-ace-key-groupcomm]. The Key Distribution Response is formatted as defined in Section 4.3.2 of [I-D.ietf-ace-key-groupcomm], with the following additions.

- * The 'key' parameter is formatted as defined in Section 6.3 of this document, with the difference that it does not include the 'group_SenderId' parameter.
- * The 'exp' parameter MUST be present.
- * The 'ace-groupcomm-profile' parameter MUST be present and has value coap_group_oscore_app.

Upon receiving the Key Distribution Response, the group member retrieves the updated security parameters and group keying material, and, if they differ from the current ones, uses them to set up the new Group OSCORE Security Context as described in Section 2 of [I-D.ietf-core-oscore-groupcomm].

9.1.2. Get Group Keying Material and OSCORE Sender ID

If the group member wants to retrieve the latest group keying material as well as the OSCORE Sender ID that it has in the OSCORE group, it sends a Key Distribution Request to the Group Manager.

In particular, it sends a CoAP GET request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME at the Group Manager.

The Group Manager processes the Key Distribution Request according to Section 4.8.1 of [I-D.ietf-ace-key-groupcomm]. The Key Distribution Response is formatted as defined in Section 4.8.1 of [I-D.ietf-ace-key-groupcomm], with the following additions.

- * The 'key' parameter is formatted as defined in Section 6.3 of this document. In particular, if the requesting group member has exclusively the role of monitor, then the 'key' parameter does not include the 'group_SenderId'.

Note that, in any other case, the current Sender ID of the group member is not specified as a separate parameter, but rather specified by 'group_SenderId' within the 'key' parameter.

- * The 'exp' parameter MUST be present.

Upon receiving the Key Distribution Response, the group member retrieves the updated security parameters, group keying material and Sender ID, and, if they differ from the current ones, uses them to set up the new Group OSCORE Security Context as described in Section 2 of [I-D.ietf-core-oscore-groupcomm].

9.2. Request to Change Individual Keying Material

As discussed in Section 2.5.2 of [I-D.ietf-core-oscore-groupcomm], a group member may at some point exhaust its Sender Sequence Numbers in the OSCORE group.

When this happens, the group member MUST send a Key Renewal Request message to the Group Manager, as per Section 4.8.2.1 of [I-D.ietf-ace-key-groupcomm]. In particular, it sends a CoAP PUT request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME at the Group Manager.

Upon receiving the Key Renewal Request, the Group Manager processes it as defined in Section 4.8.2 of [I-D.ietf-ace-key-groupcomm], with the following additions.

The Group Manager MUST return a 5.03 (Service Unavailable) response in case the OSCORE group identified by GROUPNAME is currently inactive (see Section 8.1). The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 9 ("Group currently not active").

Otherwise, the Group Manager performs one of the following actions.

1. If the requesting group member has exclusively the role of monitor, the Group Manager replies with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 1 ("Request inconsistent with the current roles").
2. Otherwise, the Group Manager takes one of the following actions.
 - * The Group Manager rekeys the OSCORE group. That is, the Group Manager generates new group keying material for that group (see Section 11), and replies to the group member with a group rekeying message as defined in Section 11, providing the new group keying material. Then, the Group Manager rekeys the rest of the OSCORE group, as discussed in Section 11.

The Group Manager SHOULD perform a group rekeying only if already scheduled to occur shortly, e.g., according to an application-specific rekeying period or scheduling, or as a reaction to a recent change in the group membership. In any other case, the Group Manager SHOULD NOT rekey the OSCORE group when receiving a Key Renewal Request (OPT12).

- * The Group Manager determines and assigns a new OSCORE Sender ID for that group member, and replies with a Key Renewal Response formatted as defined in Section 4.8.2 of [I-D.ietf-ace-key-groupcomm]. In particular, the CBOR Map in the response payload includes a single parameter 'group_SenderId' defined in Section 16.3 of this document, specifying the new Sender ID of the group member encoded as a CBOR byte string.

Consistently with Section 2.5.3.1 of [I-D.ietf-core-oscore-groupcomm], the Group Manager MUST assign a new Sender ID that has not been used in the OSCORE group since the latest time when the current Gid value was assigned to the group.

Furthermore, the Group Manager MUST add the old, relinquished Sender ID of the group member to the most recent set of stale Sender IDs, in the collection associated with the group (see Section 7.1).

The Group Manager MUST return a 5.03 (Service Unavailable) response in case there are currently no Sender IDs available to assign in the OSCORE group. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is

formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 4 ("No available node identifiers").

9.3. Retrieve Authentication Credentials of Group Members

A group member or a signature verifier may need to retrieve the authentication credentials of (other) group members. To this end, the group member or signature verifier sends a Public Key Request message to the Group Manager, as per Sections 4.4.1.1 and 4.4.2.1 of [I-D.ietf-ace-key-groupcomm]. In particular, it sends the request to the endpoint /ace-group/GROUPNAME/pub-key at the Group Manager.

If the Public Key Request uses the method FETCH, the Public Key Request is formatted as defined in Section 4.4.1 of [I-D.ietf-ace-key-groupcomm]. In particular:

- * Each element (if any) of the inner CBOR array 'role_filter' is formatted as in the inner CBOR array 'role_filter' of the 'get_pub_keys' parameter of the Joining Request when the parameter value is not the CBOR simple value "null" (0xf6) (see Section 6.1).
- * Each element (if any) of the inner CBOR array 'id_filter' is a CBOR byte string, which encodes the OSCORE Sender ID of the group member for which the associated authentication credential is requested (REQ25).

Upon receiving the Public Key Request, the Group Manager processes it as per Section 4.4.1 or Section 4.4.2 of [I-D.ietf-ace-key-groupcomm], depending on the request method being FETCH or GET, respectively. Additionally, if the Public Key Request uses the method FETCH, the Group Manager silently ignores node identifiers included in the 'get_pub_keys' parameter of the request that are not associated with any current group member (REQ26).

The success Public Key Response is formatted as defined in Section 4.4.1 or Section 4.4.2 of [I-D.ietf-ace-key-groupcomm], depending on the request method being FETCH or GET, respectively.

9.4. Upload a New Authentication Credential

A group member may need to provide the Group Manager with its new authentication credential to use in the group from then on, hence replacing the current one. This can be the case, for instance, if the signature or ECDH algorithm and possible associated parameters used in the OSCORE group have been changed, and the current authentication credential is not compatible with them.

To this end, the group member sends a Public Key Update Request message to the Group Manager, as per Section 4.9.1.1 of [I-D.ietf-ace-key-groupcomm], with the following addition.

- * The group member computes the proof-of-possession (PoP) evidence included in 'client_cred_verify' in the same way taken when preparing a Joining Request for the OSCORE group in question, as defined in Section 6.1 (REQ14).

In particular, the group member sends a CoAP POST request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME/pub-key at the Group Manager.

Upon receiving the Public Key Update Request, the Group Manager processes it as per Section 4.9.1 of [I-D.ietf-ace-key-groupcomm], with the following additions.

- * The N_S challenge used to build the proof-of-possession input is computed as defined in Section 6.1.1 (REQ15).
- * The Group Manager verifies the PoP challenge included in 'client_cred_verify' in the same way taken when processing a Joining Request for the OSCORE group in question, as defined in Section 6.2 (REQ14).
- * The Group Manager MUST return a 5.03 (Service Unavailable) response in case the OSCORE group identified by GROUPNAME is currently inactive (see Section 8.1). The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 9 ("Group currently not active").
- * If the requesting group member has exclusively the role of monitor, the Group Manager replies with a 4.00 (Bad request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 1 ("Request inconsistent with the current roles").
- * If the request is successfully processed, the Group Manager stores the association between i) the new authentication credential of the group member; and ii) the Group Identifier (Gid), i.e., the OSCORE ID Context, associated with the OSCORE group together with the OSCORE Sender ID assigned to the group member in the group. The Group Manager MUST keep this association updated over time.

9.5. Retrieve the Group Manager's Authentication Credential

A group member or a signature verifier may need to retrieve the authentication credential of the Group Manager. To this end, the requesting Client sends a KDC Public Key Request message to the Group Manager.

In particular, it sends a CoAP GET request to the endpoint `/ace-group/GROUPNAME/kdc-pub-key` at the Group Manager defined in Section 4.5.1.1 of [I-D.ietf-ace-key-groupcomm], where GROUPNAME is the name of the OSCORE group.

In addition to what is defined in Section 4.5.1 of [I-D.ietf-ace-key-groupcomm], the Group Manager MUST respond with a 4.00 (Bad Request) error response, if the requesting Client is not a current group member and GROUPNAME denotes a pairwise-only group. The response MUST have Content-Format set to `application/ace-groupcomm+cbor` and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 7 ("Signatures not used in the group").

The payload of the 2.05 (Content) KDC Public Key Response is a CBOR map, which is formatted as defined in Section 4.5.1 of [I-D.ietf-ace-key-groupcomm]. In particular, the Group Manager specifies the parameters 'kdc_cred', 'kdc_nonce' and 'kdc_challenge' as defined for the Joining Response in Section 6.3 of this document. This especially applies to the computing of the proof-of-possession (PoP) evidence included in 'kdc_cred_verify' (REQ21).

Upon receiving a 2.05 (Content) KDC Public Key Response, the requesting Client retrieves the Group Manager's authentication credential from the 'kdc_cred' parameter, and proceeds as defined in Section 4.5.1.1 of [I-D.ietf-ace-key-groupcomm]. In particular, the requesting Client verifies the PoP evidence included in 'kdc_cred_verify' by means of the same method used when processing the Joining Response, as defined in Section 6.3 of this document (REQ21).

Note that a signature verifier would not receive a successful response from the Group Manager, in case GROUPNAME denotes a pairwise-only group.

9.6. Retrieve Signature Verification Data

A signature verifier may need to retrieve data required to verify signatures of messages protected with the group mode and sent to a group (see Sections 3.1 and 8.5 of [I-D.ietf-core-oscore-groupcomm]). To this end, the signature verifier sends a Signature Verification Data Request message to the Group Manager.

In particular, it sends a CoAP GET request to the endpoint `/ace-group/GROUPNAME/verif-data` at the Group Manager defined in Section 8.2 of this document, where `GROUPNAME` is the name of the OSCORE group.

The payload of the 2.05 (Content) Signature Verification Data Response is a CBOR map, which has the format used for the Joining Response message in Section 6.3, with the following differences.

- * From the Joining Response message, only the parameters `'gkty'`, `'key'`, `'num'`, `'exp'` and `'ace-groupcomm-profile'` are present. In particular, the `'key'` parameter includes only the following data.
 - The parameters `'hkdf'`, `'contextId'`, `'cred_fmt'`, `'sign_enc_alg'`, `'sign_alg'`, `'sign_params'`. These parameters MUST be present.
 - The parameters `'alg'` and `'ecdh_alg'`. These parameter MUST NOT be present if the group is a signature-only group. Otherwise, they MUST be present.
- * The parameter `'group_enc_key'` is also included, with CBOR label defined in Section 16.3. This parameter specifies the Group Encryption Key of the OSCORE Group, encoded as a CBOR byte string. The Group Manager derives the Group Encryption Key from the group keying material, as per Section 2.1.6 of [I-D.ietf-core-oscore-groupcomm]. This parameter MUST be present.

In order to verify signatures in the group (see Section 8.5 of [I-D.ietf-core-oscore-groupcomm]), the signature verifier relies on: the data retrieved from the 2.05 (Content) Signature Verification Data Response; the public keys of the group members signing the messages to verify, retrieved from those members' authentication credentials that can be obtained as defined in Section 9.3; and the public key of the Group Manager, retrieved from the Group Manager's authentication credential that can be obtained as defined in Section 9.5.

Figure 3 gives an overview of the exchange described above, while Figure 4 shows an example.

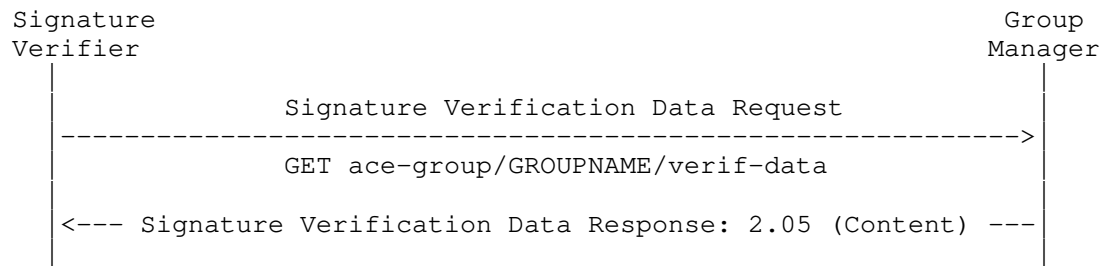


Figure 3: Message Flow of Signature Verification Data Request-Response

Request:

```

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "verif-data"
Payload: -
  
```

Response:

```

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation, with GROUPCOMM_KEY_TBD
        and PROFILE_TBD being CBOR integers, while GROUP_ENC_KEY
        being a CBOR byte string):
{
  "gkty": GROUPCOMM_KEY_TBD,
  "key": {
    'hkdf': 5,                                ; HMAC 256/256
    'contextId': h'37fc',
    'cred_fmt': 33,                            ; x5chain
    'sign_enc_alg': 10,                        ; AES-CCM-16-64-128
    'sign_alg': -8,                            ; EdDSA
    'sign_params': [[1], [1, 6]]              ; [[OKP], [OKP, Ed25519]]
  },
  "num": 12,
  "exp": 1609459200,
  "ace_groupcomm_profile": PROFILE_TBD,
  "group_enc_key": GROUP_ENC_KEY
}
  
```

Figure 4: Example of Signature Verification Data Request-Response

9.7. Retrieve the Group Policies

A group member may request the current policies used in the OSCORE group. To this end, the group member sends a Policies Request, as per Section 4.6.1.1 of [I-D.ietf-ace-key-groupcomm]. In particular, it sends a CoAP GET request to the endpoint `/ace-group/GROUPNAME/policies` at the Group Manager, where GROUPNAME is the name of the OSCORE group.

Upon receiving the Policies Request, the Group Manager processes it as per Section 4.6.1 of [I-D.ietf-ace-key-groupcomm]. The success Policies Response is formatted as defined in Section 4.6.1 of [I-D.ietf-ace-key-groupcomm].

9.8. Retrieve the Keying Material Version

A group member may request the current version of the keying material used in the OSCORE group. To this end, the group member sends a Version Request, as per Section 4.7.1.1 of [I-D.ietf-ace-key-groupcomm]. In particular, it sends a CoAP GET request to the endpoint `/ace-group/GROUPNAME/num` at the Group Manager, where GROUPNAME is the name of the OSCORE group.

Upon receiving the Version Request, the Group Manager processes it as per Section 4.7.1 of [I-D.ietf-ace-key-groupcomm]. The success Version Response is formatted as defined in Section 4.7.1 of [I-D.ietf-ace-key-groupcomm].

9.9. Retrieve the Group Status

A group member may request the current status of the the OSCORE group, i.e., active or inactive. To this end, the group member sends a Group Status Request to the Group Manager.

In particular, the group member sends a CoAP GET request to the endpoint `/ace-group/GROUPNAME/active` at the Group Manager defined in Section 8.1 of this document, where GROUPNAME is the name of the OSCORE group.

The payload of the 2.05 (Content) Group Status Response includes the CBOR simple value "true" (0xf5) if the group is currently active, or the CBOR simple value "false" (0xf4) otherwise. The group is considered active if it is set to allow new members to join, and if communication within the group is fine to happen.

Upon learning from a 2.05 (Content) response that the group is currently inactive, the group member SHOULD stop taking part in communications within the group, until it becomes active again.

Upon learning from a 2.05 (Content) response that the group has become active again, the group member can resume taking part in communications within the group.

Figure 5 gives an overview of the exchange described above, while Figure 6 shows an example.

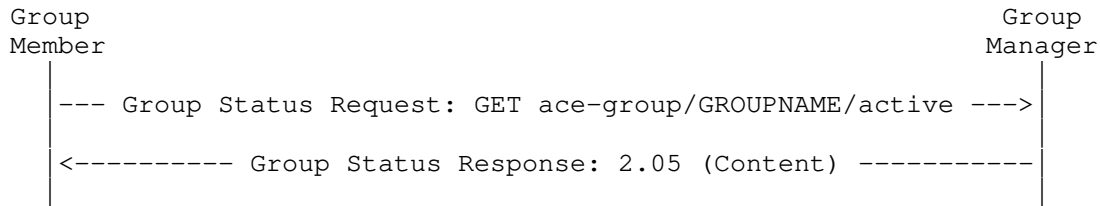


Figure 5: Message Flow of Group Status Request-Response

Request:

```

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "active"
Payload: -
  
```

Response:

```

Header: Content (Code=2.05)
Payload (in CBOR diagnostic notation):
  true
  
```

Figure 6: Example of Group Status Request-Response

9.10. Retrieve Group Names

A node may want to retrieve from the Group Manager the group name and the URI of the group-membership resource of a group. This is relevant in the following cases.

- * Before joining a group, a joining node may know only the current Group Identifier (Gid) of that group, but not the group name and the URI to the group-membership resource.
- * As current group member in several groups, the node has missed a previous group rekeying in one of them (see Section 11). Hence, it retains stale keying material and fails to decrypt received messages exchanged in that group.

Such messages do not provide a direct hint to the correct group name, that the node would need in order to retrieve the latest keying material and authentication credentials from the Group Manager (see Section 9.1.1, Section 9.1.2 and Section 9.3). However, such messages may specify the current Gid of the group, as value of the 'kid_context' field of the OSCORE CoAP option (see Section 6.1 of [RFC8613] and Section 4.2 of [I-D.ietf-core-oscore-groupcomm]).

- * As signature verifier, the node also refers to a group name for retrieving the required authentication credentials from the Group Manager (see Section 9.3). As discussed above, intercepted messages do not provide a direct hint to the correct group name, while they may specify the current Gid of the group, as value of the 'kid_context' field of the OSCORE CoAP option. In such a case, upon intercepting a message in the group, the node requires to correctly map the Gid currently used in the group with the invariant group name.

Furthermore, since it is not a group member, the node does not take part to a possible group rekeying. Thus, following a group rekeying and the consequent change of Gid in a group, the node would retain the old Gid value and cannot correctly associate intercepted messages to the right group, especially if acting as signature verifier in several groups. This in turn prevents the efficient verification of signatures, and especially the retrieval of required, new authentication credentials from the Group Manager.

In either case, the node only knows the current Gid of the group, as learned from received or intercepted messages exchanged in the group. As detailed below, the node can contact the Group Manager, and request the group name and URI to the group-membership resource corresponding to that Gid. Then, it can use that information to either join the group as a candidate group member, get the latest keying material as a current group member, or retrieve authentication credentials used in the group as a signature verifier. To this end, the node sends a Group Name and URI Retrieval Request, as per Section 4.2.1.1 of [I-D.ietf-ace-key-groupcomm].

In particular, the node sends a CoAP FETCH request to the endpoint /ace-group at the Group Manager formatted as defined in Section 4.2.1 of [I-D.ietf-ace-key-groupcomm]. Each element of the CBOR array 'gid' is a CBOR byte string (REQ13), which encodes the Gid of the group for which the group name and the URI to the group-membership resource are requested.

Upon receiving the Group Name and URI Retrieval Request, the Group Manager processes it as per Section 4.2.1 of [I-D.ietf-ace-key-groupcomm]. The success Group Name and URI Retrieval Response is formatted as defined in Section 4.2.1 of [I-D.ietf-ace-key-groupcomm]. In particular, each element of the CBOR array 'gid' is a CBOR byte string (REQ13), which encodes the Gid of the group for which the group name and the URI to the group-membership resource are provided.

For each of its groups, the Group Manager maintains an association between the group name and the URI to the group-membership resource on one hand, and only the current Gid for that group on the other hand. That is, the Group Manager does not maintain an association between the former pair and any other Gid for that group than the current, most recent one.

Figure 7 gives an overview of the exchanges described above, while Figure 8 shows an example.

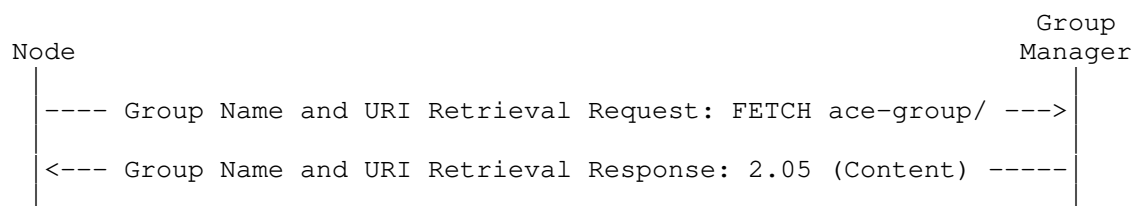


Figure 7: Message Flow of Group Name and URI Retrieval Request-Response

Request:

```
Header: FETCH (Code=0.05)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
{
  "gid": [h'37fc', h'84bd']
}
```

Response:

```
Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
{
  "gid": [h'37fc', h'84bd'],
  "gname": ["g1", "g2"],
  "guri": ["ace-group/g1", "ace-group/g2"]
}
```

Figure 8: Example of Group Name and URI Retrieval Request-Response

9.11. Leave the Group

A group member may request to leave the OSCORE group. To this end, the group member sends a Group Leaving Request, as per Section 4.8.3.1 of [I-D.ietf-ace-key-groupcomm]. In particular, it sends a CoAP DELETE request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME at the Group Manager.

Upon receiving the Group Leaving Request, the Group Manager processes it as per Section 4.8.3 of [I-D.ietf-ace-key-groupcomm]. Then, the Group Manager performs the follow-up actions defined in Section 10 of this document.

10. Removal of a Group Member

Other than after a spontaneous request to the Group Manager as described in Section 9.11, a node may be forcibly removed from the OSCORE group, e.g., due to expired or revoked authorization.

In either case, if the Group Manager reassigns Gid values during the group's lifetime (see Section 3.2.1.1 of [I-D.ietf-core-oscore-groupcomm]), the Group Manager "forgets" the Birth Gid currently associated with the leaving node in the OSCORE group. This was stored following the Joining Response sent to that node, after its latest (re-)joining of the OSCORE group (see Section 6.3).

If any of the two conditions below holds, the Group Manager MUST inform the leaving node of its eviction as follows. If both conditions hold, the Group Manager MUST inform the leaving node by using only the method corresponding to one of either conditions.

- * If, upon joining the group (see Section 6.1), the leaving node specified a URI in the 'control_uri' parameter defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], the Group Manager sends a DELETE request targeting the URI specified in the 'control_uri' parameter (OPT7).
- * If the leaving node has been observing the associated resource at ace-group/GROUPNAME/nodes/NODENAME, the Group Manager sends an unsolicited 4.04 (Not Found) error response to the leaving node, as specified in Section 4.3.2 of [I-D.ietf-ace-key-groupcomm].

Furthermore, the Group Manager might intend to evict all the current group members from the group at once. In such a case, if the Joining Responses sent by the Group Manager to nodes joining the group (see Section 6.3) specify a URI in the 'control_group_uri' parameter defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], then the Group Manager MUST additionally send a DELETE request targeting the URI specified in the 'control_group_uri' parameter (OPT10).

If the leaving node has not exclusively the role of monitor, the Group Manager performs the following actions.

- * The Group Manager frees the OSCORE Sender ID value of the leaving node. This value MUST NOT become available for possible upcoming joining nodes in the same group, until the group has been rekeyed and assigned a new Group Identifier (Gid).
- * The Group Manager MUST add the relinquished Sender ID of the leaving node to the most recent set of stale Sender IDs, in the collection associated with the group (see Section 7.1).
- * The Group Manager cancels the association between, on one hand, the authentication credential of the leaving node and, on the other hand, the Gid associated with the OSCORE group together with the freed Sender ID value. The Group Manager deletes the

authentication credential of the leaving node, if that authentication credential has no remaining association with any pair (Gid, Sender ID).

Then, the Group Manager MUST generate updated security parameters and group keying material, and provide it to the remaining group members (see Section 11). As a consequence, the leaving node is not able to acquire the new security parameters and group keying material distributed after its leaving.

The same considerations from Section 5 of [I-D.ietf-ace-key-groupcomm] apply here as well, considering the Group Manager acting as KDC.

11. Group Rekeying Process

In order to rekey the OSCORE group, the Group Manager distributes a new Group Identifier (Gid), i.e., a new OSCORE ID Context; a new OSCORE Master Secret; and, optionally, a new OSCORE Master Salt for that group. When doing so, the Group Manager MUST increment the version number of the group keying material, before starting its distribution.

As per Section 3.2.1.1 of [I-D.ietf-core-oscore-groupcomm], the Group Manager MAY reassign a Gid to the same group over that group's lifetime, e.g., once the whole space of Gid values has been used for the group in question. If the Group Manager supports reassignment of Gid values and performs it in a group, then the Group Manager additionally takes the following actions.

- * Before rekeying the group, the Group Manager MUST check if the new Gid to be distributed coincides with the Birth Gid of any of the current group members (see Section 6.3).
- * If any of such "elder members" is found in the group, the Group Manager MUST evict them from the group. That is, the Group Manager MUST terminate their membership and MUST rekey the group in such a way that the new keying material is not provided to those evicted elder members. This also includes adding their relinquished Sender IDs to the most recent set of stale Sender IDs, in the collection associated with the group (see Section 7.1), before rekeying the group.

Until a further following group rekeying, the Group Manager MUST store the list of those latest-evicted elder members. If any of those nodes re-joins the group before a further following group rekeying occurs, the Group Manager MUST NOT rekey the group upon their re-joining. When one of those nodes re-joins the group, the Group Manager can rely, e.g., on the ongoing secure communication association to recognize the node as included in the stored list.

Across the rekeying execution, the Group Manager MUST preserve the same unchanged OSCORE Sender IDs for all group members intended to remain in the group. This avoids affecting the retrieval of authentication credentials from the Group Manager and the verification of group messages.

The Group Manager MUST support the "Point-to-Point" group rekeying scheme registered in Section 11.14 of [I-D.ietf-ace-key-groupcomm], as per the detailed use defined in Section 11.1 of this document. Future specifications may define how this application profile can use alternative group rekeying schemes, which MUST comply with the functional steps defined in Section 3.2 of [I-D.ietf-core-oscore-groupcomm]. The Group Manager MUST indicate the use of such an alternative group rekeying scheme to joining nodes, by means of the 'group_rekeying' parameter included in Joining Response messages (see Section 6.3).

It is RECOMMENDED that the Group Manager gets confirmation of successful distribution from the group members, and admits a maximum number of individual retransmissions to non-confirming group members. Once completed the group rekeying process, the Group Manager creates a new empty set X' of stale Sender IDs associated with the version of the newly distributed group keying material. Then, the Group Manager MUST add the set X' to the collection of stale Sender IDs associated with the group (see Section 7.1).

In case the rekeying terminates and some group members have not received the new keying material, they will not be able to correctly process following secured messages exchanged in the group. These group members will eventually contact the Group Manager, in order to retrieve the current keying material and its version.

Some of these group members may be in multiple groups, each associated with a different Group Manager. When failing to correctly process messages secured with the new keying material, these group members may not have sufficient information to determine which exact Group Manager they should contact, in order to retrieve the current keying material they are missing.

If the Gid is formatted as described in Appendix C of [I-D.ietf-core-oscore-groupcomm], the Group Prefix can be used as a hint to determine the right Group Manager, as long as no collisions among Group Prefixes are experienced. Otherwise, a group member needs to contact the Group Manager of each group, e.g., by first requesting only the version of the current group keying material (see Section 9.8) and then possibly requesting the current keying material (see Section 9.1.1).

Furthermore, some of these group members can be in multiple groups, all of which associated with the same Group Manager. In this case, these group members may also not have sufficient information to determine which exact group they should refer to, when contacting the right Group Manager. Hence, they need to contact a Group Manager multiple times, i.e., separately for each group they belong to and associated with that Group Manager.

Section 11.2 defines the actions performed by a group member upon receiving the new group keying material. Section 11.3 discusses how a group member can realize that it has missed one or more rekeying instances, and the actions it is accordingly required to take.

11.1. Sending Rekeying Messages

When using the "Point-to-Point" group rekeying scheme, the group rekeying messages MUST have Content-Format set to application/ace-groupcomm+cbor and have the same format used for the Joining Response message in Section 6.3, with the following differences. Note that this extends the minimal content of a rekeying message as defined in Section 6 of [I-D.ietf-ace-key-groupcomm] (OPT14).

- * From the Joining Response, only the parameters 'gkty', 'key', 'num', 'exp', and 'ace-groupcomm-profile' are present. In particular, the 'key' parameter includes only the following data.
 - The 'ms' parameter, specifying the new OSCORE Master Secret value. This parameter MUST be present.
 - The 'contextId' parameter, specifying the new Gid to use as OSCORE ID Context value. This parameter MUST be present.
 - The 'salt' value, specifying the new OSCORE Master Salt value. This parameter MAY be present.

- * The parameter 'stale_node_ids' MUST also be included, with CBOR label defined in Section 16.3. This parameter is encoded as a CBOR array, where each element is encoded as a CBOR byte string. The CBOR array has to be intended as a set, i.e., the order of its elements is irrelevant. The parameter is populated as follows.
 - The Group Manager creates an empty CBOR array ARRAY.
 - The Group Manager considers the collection of stale Sender IDs associated with the group (see Section 7.1), and takes the most recent set X, i.e., the set associated with the current version of the group keying material about to be relinquished.
 - For each Sender ID in X, the Group Manager encodes it as a CBOR byte string and adds the result to ARRAY.
 - The parameter 'stale_node_ids' takes ARRAY as value.
- * The parameters 'pub_keys', 'peer_roles' and 'peer_identifiers' SHOULD be present, if the group rekeying is performed due to one or multiple Clients that have requested to join the group. Following the same semantics used in the Joining Response message (see Section 6.3), the three parameters specify the authentication credential, roles in the group and node identifier of each of the Clients that have requested to join the group. The Group Manager MUST NOT include a non-empty subset of these three parameters.

The Group Manager separately sends a group rekeying message formatted as defined above to each group member to be rekeyed.

Each rekeying message MUST be secured with the pairwise secure communication association between the Group Manager and the group member used during the joining process. In particular, each rekeying message can target the 'control_uri' URI path defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm] (OPT7), if provided by the intended recipient upon joining the group (see Section 6.1).

This distribution approach requires group members to act (also) as servers, in order to correctly handle unsolicited group rekeying messages from the Group Manager. In particular, if a group member and the Group Manager use OSCORE [RFC8613] to secure their pairwise communications, the group member MUST create a Replay Window in its own Recipient Context upon establishing the OSCORE Security Context with the Group Manager, e.g., by means of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

Group members and the Group Manager SHOULD additionally support alternative distribution approaches that do not require group members to act (also) as servers. A number of such approaches are defined in Section 6 of [I-D.ietf-ace-key-groupcomm]. In particular, a group member may use CoAP Observe [RFC7641] and subscribe for updates to the group-membership resource of the group, at the endpoint /ace-group/GROUPNAME/ of the Group Manager (see Section 6.1 of [I-D.ietf-ace-key-groupcomm]). Alternatively, a full-fledged Pub-Sub model can be considered [I-D.ietf-core-coap-pubsub], where the Group Manager publishes to a rekeying topic hosted at a Broker, while the group members subscribe to such topic (see Section 6.2 of [I-D.ietf-ace-key-groupcomm]).

11.2. Receiving Rekeying Messages

Once received the new group keying material, a group member proceeds as follows. Unless otherwise specified, the following is independent of the specifically used group rekeying scheme.

The group member considers the stale Sender IDs received from the Group Manager. If the "Point-to-Point" group rekeying scheme as detailed in Section 11.1 is used, the stale Sender IDs are specified by the 'stale_node_ids' parameter.

After that, as per Section 3.2 of [I-D.ietf-core-oscore-groupcomm], the group member MUST remove every authentication credential associated with a stale Sender ID from its list of group members' authentication credentials used in the group, and MUST delete each of its Recipient Contexts used in the group whose corresponding Recipient ID is a stale Sender ID.

Then, the following cases can occur, based on the version number V' of the new group keying material distributed through the rekeying process. If the "Point-to-Point" group rekeying scheme as detailed in Section 11.1 is used, this information is specified by the 'num' parameter.

- * The group member has not missed any group rekeying. That is, the old keying material stored by the group member has version number V , while the received new keying material has version number $V' = (V + 1)$. In such a case, the group member simply installs the new keying material and derives the corresponding new Security Context.

- * The group member has missed one or more group rekeying instances. That is, the old keying material stored by the group member has version number V , while the received new keying material has version number $V' > (V + 1)$. In such a case, the group member MUST proceed as defined in Section 11.3.
- * The group member has received keying material not newer than the stored one. That is, the old keying material stored by the group member has version number V , while the received keying material has version number $V' < (V + 1)$. In such a case, the group member MUST ignore the received rekeying messages and MUST NOT install the received keying material.

11.3. Missed Rekeying Instances

A group member can realize to have missed one or more rekeying instances in one of the ways discussed below. In the following, V denotes the version number of the old keying material stored by the group member, while V' denotes the version number of the latest, possibly just distributed, keying material.

- a. The group member has participated to a rekeying process that has distributed new keying material with version number $V' > (V + 1)$, as discussed in Section 11.2.
- b. The group member has obtained the latest keying material from the Group Manager, as a response to a Key Distribution Request (see Section 9.1.1) or to a Joining Request when re-joining the group (see Section 6.1). In particular, V is different than V' specified by the 'num' parameter in the response.
- c. The group member has obtained the authentication credentials of other group members, through a Public Key Request-Response exchange with the Group Manager (see Section 9.3). In particular, V is different than V' specified by the 'num' parameter in the response.
- d. The group member has performed a Version Request-Response exchange with the Group Manager (see Section 9.8). In particular, V is different than V' specified by the 'num' parameter in the response.

In either case, the group member MUST delete the stored keying material with version number V .

If case (a) or case (b) applies, the group member MUST perform the following actions.

1. The group member MUST NOT install the latest keying material yet, in case that was already obtained.
2. The group member sends a Stale Sender IDs Request to the Group Manager (see Section 11.3.1), specifying the version number V as payload of the request.

If the Stale Sender IDs Response from the Group Manager has no payload, the group member MUST remove all the authentication credentials from its list of group members' authentication credentials used in the group, and MUST delete all its Recipient Contexts used in the group.

Otherwise, the group member considers the stale Sender IDs specified in the Stale Sender IDs Response from the Group Manager. Then, the group member MUST remove every authentication credential associated with a stale Sender ID from its list of group members' authentication credentials used in the group, and MUST delete each of its Recipient Contexts used in the group whose corresponding Recipient ID is a stale Sender ID.

3. The group member installs the latest keying material with version number V' and derives the corresponding new Security Context.

If case (c) or case (d) applies, the group member SHOULD perform the following actions.

1. The group member sends a Stale Sender IDs Request to the Group Manager (see Section 11.3.1), specifying the version number V as payload of the request.

If the Stale Sender IDs Response from the Group Manager has no payload, the group member MUST remove all the authentication credentials from its list of group members' authentication credentials used in the group, and MUST delete all its Recipient Contexts used in the group.

Otherwise, the group member considers the stale Sender IDs specified in the Stale Sender IDs Response from the Group Manager. Then, the group member MUST remove every authentication credential associated with a stale Sender ID from its list of group members' authentication credentials used in the group, and MUST delete each of its Recipient Contexts used in the group whose corresponding Recipient ID is a stale Sender ID.

2. The group member obtains the latest keying material with version number V' from the Group Manager. This can happen by sending a Key Distribution Request to the Group Manager (see Section 9.1.1) and Section 9.1.2).
3. The group member installs the latest keying material with version number V' and derives the corresponding new Security Context.

If case (c) or case (d) applies, the group member can alternatively perform the following actions.

1. The group member re-joins the group (see Section 6.1). When doing so, the group member MUST re-join with the same roles it currently has in the group, and MUST request the Group Manager for the authentication credentials of all the current group members. That is, the 'get_pub_keys' parameter of the Joining Request MUST be present and MUST be set to the CBOR simple value "null" (0xf6).
2. When receiving the Joining Response (see Section 6.4 and Section 6.4), the group member retrieves the set Z of authentication credentials specified in the 'pub_keys' parameter.

Then, the group member MUST remove every authentication credential which is not in Z from its list of group members' authentication credentials used in the group, and MUST delete each of its Recipient Contexts used in the group that does not include any of the authentication credentials in Z .

3. The group member installs the latest keying material with version number V' and derives the corresponding new Security Context.

11.3.1. Retrieve Stale Sender IDs

When realizing to have missed one or more group rekeying instances (see Section 11.3), a node needs to retrieve from the Group Manager the data required to delete some of its stored group members' authentication credentials and Recipient Contexts (see Section 8.3.1). These data are provided as an aggregated set of stale Sender IDs, which are used as specified in Section 11.3.

In particular, the node sends a CoAP FETCH request to the endpoint /ace-group/GROUPNAME/stale-sids at the Group Manager defined in Section 8.3 of this document, where GROUPNAME is the name of the OSCORE group.

The payload of the Stale Sender IDs Request MUST include a CBOR unsigned integer. This encodes the version number V of the most recent group keying material stored and installed by the requesting Client, which is older than the latest, possibly just distributed, keying material with version number V' .

The handler MUST reply with a 4.00 (Bad Request) error response, if the request is not formatted correctly. Also, the handler MUST respond with a 4.00 (Bad Request) error response, if the specified version number V is greater or equal than the version number V' associated with the latest keying material in the group, i.e., in case $V \geq V'$.

Otherwise, the handler responds with a 2.05 (Content) Stale Sender IDs Response. The payload of the response is formatted as defined below, where $SKEW = (V' - V + 1)$.

- * The Group Manager considers ITEMS as the current number of sets stored in the collection of stale Sender IDs associated with the group (see Section 7.1).
- * If $SKEW > ITEMS$, the Stale Sender IDs Response MUST NOT have a payload.
- * Otherwise, the payload of the Stale Sender IDs Response MUST include a CBOR array, where each element is encoded as a CBOR byte string. The CBOR array has to be intended as a set, i.e., the order of its elements is irrelevant. The Group Manager populates the CBOR array as follows.
 - The Group Manager creates an empty CBOR array ARRAY and an empty set X.
 - The Group Manager considers the SKEW most recent sets stored in the collection of stale Sender IDs associated with the group. Note that the most recent set is the one associate to the latest version of the group keying material.
 - The Group Manager copies all the Sender IDs from the selected sets into X. When doing so, the Group Manager MUST discard duplicates. That is, the same Sender ID MUST NOT be present more than once in the final content of X.
 - For each Sender ID in X, the Group Manager encodes it as a CBOR byte string and adds the result to ARRAY.

- Finally, ARRAY is specified as payload of the Stale Sender IDs Response. Note that ARRAY might result in the empty CBOR array.

Figure 9 gives an overview of the exchange described above, while Figure 10 shows an example.

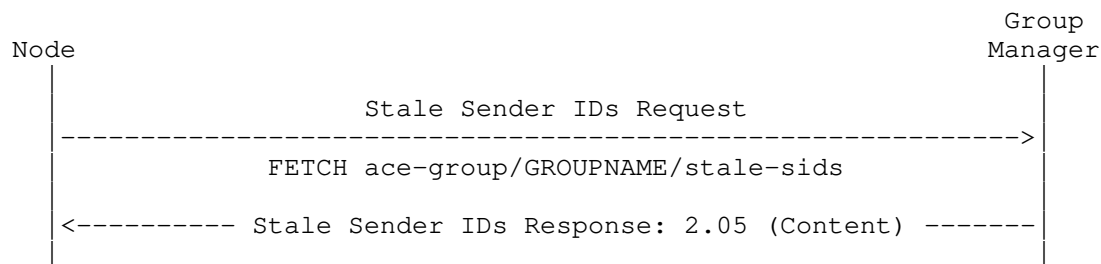


Figure 9: Message Flow of Stale Sender IDs Request-Response

Request:

```

Header: FETCH (Code=0.05)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "stale-sids"
Payload (in CBOR diagnostic notation):
  42
  
```

Response:

```

Header: Content (Code=2.05)
Payload (in CBOR diagnostic notation):
  [h'01', h'fc', h'12ab', h'de44', h'ff']
  
```

Figure 10: Example of Stale Sender IDs Request-Response

12. ACE Groupcomm Parameters

In addition to those defined in Section 8 of [I-D.ietf-ace-key-groupcomm], this application profile defines additional parameters used during the second part of the message exchange with the Group Manager, i.e., after the exchange of Token Transfer Request and Response (see Section 5.3). The table below summarizes them and specifies the CBOR key to use instead of the full descriptive name.

Note that the media type `application/ace-groupcomm+cbor` MUST be used when these parameters are transported in the respective message fields.

Name	CBOR Key	CBOR Type	Reference
<code>group_senderId</code>	TBD	<code>bstr</code>	[this document]
<code>ecdh_info</code>	TBD	<code>array</code>	[this document]
<code>kdc_dh_creds</code>	TBD	<code>array</code>	[this document]
<code>group_enc_key</code>	TBD	<code>bstr</code>	[this document]
<code>stale_node_ids</code>	TBD	<code>array</code>	[this document]

Figure 11: ACE Groupcomm Parameters

The Group Manager is expected to support and understand all the parameters above. Instead, a Client is required to support the new parameters defined in this application profile as specified below (REQ29).

- * `'group_senderId'` MUST be supported by a Client that intends to join an OSCORE group with the role of Requester and/or Responder.
- * `'ecdh_info'` MUST be supported by a Client that intends to join a group which uses the pairwise mode of Group OSCORE.
- * `'kdc_dh_creds'` MUST be supported by a Client that intends to join a group which uses the pairwise mode of Group OSCORE and that does not plan to or cannot rely on an early retrieval of the Group Manager's Diffie-Hellman authentication credential.
- * `'group_enc_key'` MUST be supported by a Client that intends to join a group which uses the group mode of Group OSCORE or to be signature verifier for that group.
- * `'stale_node_ids'` MUST be supported.

When the conditional parameters defined in Section 8 of [I-D.ietf-ace-key-groupcomm] are used with this application profile, a Client must, should or may support them as specified below (REQ30).

- * `'client_cred', 'cnonce', 'client_cred_verify'`. A Client that has an own authentication credential to use in a group MUST support these parameters.
- * `'kdcchallenge'`. A Client that has an own authentication credential to use in a group and that provides the Access Token to the Group Manager through a Token Transfer Request (see Section 5.3) MUST support this parameter.
- * `'pub_keys_repo'`. This parameter is not relevant for this application profile, since the Group Manager always acts as repository of the group members' authentication credentials.
- * `'group_policies'`. A Client that is interested in the specific policies used in a group, but that does not know them or cannot become aware of them before joining that group, SHOULD support this parameter.
- * `'peer_roles'`. A Client MUST support this parameter, since in this application profile it is relevant for Clients to know the roles of the group member associated with each authentication credential.
- * `'kdc_nonce', 'kdc_cred' and 'kdc_cred_verify'`. A Client MUST support these parameters, since the Group Manager's authentication credential is required to process messages protected with Group OSCORE (see Section 4.3 of [I-D.ietf-core-oscore-groupcomm]).
- * `'mgt_key_material'`. A Client that supports an advanced rekeying scheme possibly used in the group, such as based on one-to-many rekeying messages sent by the Group Manager (e.g., over IP multicast), MUST support this parameter.
- * `'control_group_uri'`. A Client that supports the hosting of local resources each associated with a group (hence acting as CoAP server) and the reception of one-to-many requests sent to those resources by the Group Manager (e.g., over IP multicast) MUST support this parameter.

13. ACE Groupcomm Error Identifiers

In addition to those defined in Section 9 of [I-D.ietf-ace-key-groupcomm], this application profile defines new values that the Group Manager can include as error identifiers, in the `'error'` field of an error response with Content-Format `application/ace-groupcomm+cbor`.

Value	Description
7	Signatures not used in the group
8	Operation permitted only to signature verifiers
9	Group currently not active

Figure 12: ACE Groupcomm Error Identifiers

A Client supporting the 'error' parameter (see Sections 4.1.2 and 8 of [I-D.ietf-ace-key-groupcomm]) and able to understand the specified error may use that information to determine what actions to take next. If it is included in the error response and supported by the Client, the 'error_description' parameter may provide additional context. In particular, the following guidelines apply.

- * In case of error 7, the Client should stop sending the request in question to the Group Manager. In this application profile, this error is relevant only for a signature verifier, in case it tries to access resources related to a pairwise-only group.
- * In case of error 8, the Client should stop sending the request in question to the Group Manager.
- * In case of error 9, the Client should wait for a certain (pre-configured) amount of time, before trying re-sending its request to the Group Manager.

14. Default Values for Group Configuration Parameters

This section defines the default values that the Group Manager assumes for the configuration parameters of an OSCORE group, unless differently specified when creating and configuring the group. This can be achieved as specified in [I-D.ietf-ace-oscore-gm-admin].

14.1. Common

This section always applies, as related to common configuration parameters.

- * For the HKDF Algorithm 'hkdf', the Group Manager SHOULD use HKDF SHA-256, defined as default in Section 3.2 of [RFC8613]. In the 'hkdf' parameter, this HKDF Algorithm is specified by the HMAC Algorithm HMAC 256/256 (COSE algorithm encoding: 5).

- * For the format 'cred_fmt' used for the authentication credentials in the group, the Group Manager SHOULD use CBOR Web Token (CWT) or CWT Claims Set (CCS) [RFC8392], i.e., the COSE Header Parameter 'kcwt' and 'kccs', respectively.

[These COSE Header Parameters are under pending registration requested by draft-ietf-lake-edhoc.]

- * For 'max_stale_sets', the Group Manager SHOULD consider $N = 3$ as the maximum number of stored sets of stale Sender IDs in the collection associated with the group (see Section 7.1).

14.2. Group Mode

This section applies if the group uses (also) the group mode of Group OSCORE.

- * For the Signature Encryption Algorithm 'sign_enc_alg' used to encrypt messages protected with the group mode, the Group Manager SHOULD use AES-CCM-16-64-128 (COSE algorithm encoding: 10) as default value.

The Group Manager SHOULD use the following default values for the Signature Algorithm 'sign_alg' and related parameters 'sign_params', consistently with the "COSE Algorithms" registry [COSE.Algorithms], the "COSE Key Types" registry [COSE.Key.Types] and the "COSE Elliptic Curves" registry [COSE.Elliptic.Curves].

- * For the Signature Algorithm 'sign_alg' used to sign messages protected with the group mode, the signature algorithm EdDSA [RFC8032].
- * For the parameters 'sign_params' of the Signature Algorithm:
 - The array [[OKP], [OKP, Ed25519]], in case EdDSA is assumed or specified for 'sign_alg'. In particular, this indicates to use the COSE key type OKP and the elliptic curve Ed25519 [RFC8032].
 - The array [[EC2], [EC2, P-256]], in case ES256 [RFC6979] is specified for 'sign_alg'. In particular, this indicates to use the COSE key type EC2 and the elliptic curve P-256.
 - The array [[EC2], [EC2, P-384]], in case ES384 [RFC6979] is specified for 'sign_alg'. In particular, this indicates to use the COSE key type EC2 and the elliptic curve P-384.

- The array `[[EC2], [EC2, P-521]]`, in case ES512 [RFC6979] is specified for `'sign_alg'`. In particular, this indicates to use the COSE key type EC2 and the elliptic curve P-521.
- The array `[[RSA], [RSA]]`, in case PS256, PS384 or PS512 [RFC8017] is specified for `'sign_alg'`. In particular, this indicates to use the COSE key type RSA.

14.3. Pairwise Mode

This section applies if the group uses (also) the pairwise mode of Group OSCORE.

For the AEAD Algorithm `'alg'` used to encrypt messages protected with the pairwise mode, the Group Manager SHOULD use the same default value defined in Section 3.2 of [RFC8613], i.e., AES-CCM-16-64-128 (COSE algorithm encoding: 10).

For the Pairwise Key Agreement Algorithm `'ecdh_alg'` and related parameters `'ecdh_params'`, the Group Manager SHOULD use the following default values, consistently with the "COSE Algorithms" registry [COSE.Algorithms], the "COSE Key Types" registry [COSE.Key.Types] and the "COSE Elliptic Curves" registry [COSE.Elliptic.Curves].

- * For the Pairwise Key Agreement Algorithm `'ecdh_alg'` used to compute static-static Diffie-Hellman shared secrets, the ECDH algorithm ECDH-SS + HKDF-256 specified in Section 6.3.1 of [I-D.ietf-cose-rfc8152bis-algs].
- * For the parameters `'ecdh_params'` of the Pairwise Key Agreement Algorithm:
 - The array `[[OKP], [OKP, X25519]]`, in case EdDSA is assumed or specified for `'sign_alg'`, or in case the group is a pairwise-only group. In particular, this indicates to use the COSE key type OKP and the elliptic curve X25519 [RFC8032].
 - The array `[[EC2], [EC2, P-256]]`, in case ES256 [RFC6979] is specified for `'sign_alg'`. In particular, this indicates to use the COSE key type EC2 and the elliptic curve P-256.
 - The array `[[EC2], [EC2, P-384]]`, in case ES384 [RFC6979] is specified for `'sign_alg'`. In particular, this indicates to use the COSE key type EC2 and the elliptic curve P-384.
 - The array `[[EC2], [EC2, P-521]]`, in case ES512 [RFC6979] is specified for `'sign_alg'`. In particular, this indicates to use the COSE key type EC2 and the elliptic curve P-521.

15. Security Considerations

Security considerations for this profile are inherited from [I-D.ietf-ace-key-groupcomm], the ACE framework for Authentication and Authorization [I-D.ietf-ace-oauth-authz], and the specific transport profile of ACE signalled by the AS, such as [I-D.ietf-ace-dtls-authorize] and [I-D.ietf-ace-oscore-profile].

The following security considerations also apply for this profile.

15.1. Management of OSCORE Groups

This profile leverages the following management aspects related to OSCORE groups and discussed in the sections of [I-D.ietf-core-oscore-groupcomm] referred below.

- * Management of group keying material (see Section 3.2 of [I-D.ietf-core-oscore-groupcomm]). The Group Manager is responsible for the renewal and re-distribution of the keying material in the groups of its competence (rekeying).

The Group Manager performs a rekeying when one or more members leave the group, thus preserving forward security and ensuring that the security properties of Group OSCORE are fulfilled. According to the specific application requirements, the Group Manager can also rekey the group upon a new node's joining, in case backward security has also to be preserved.

- * Provisioning and retrieval of authentication credentials (see Section 3 of [I-D.ietf-core-oscore-groupcomm]). The Group Manager acts as repository of authentication credentials of group members, and provides them upon request.
- * Synchronization of sequence numbers (see Section 6.3 of [I-D.ietf-core-oscore-groupcomm]). This concerns how a responder node that has just joined an OSCORE group can synchronize with the sequence number of requesters in the same group.

Before sending the Joining Response, the Group Manager MUST verify that the joining node actually owns the associated private key. To this end, the Group Manager can rely on the proof-of-possession challenge-response defined in Section 6.

Alternatively, when establishing a secure communication association with the Group Manager, the joining node can provide the Group Manager with its own authentication credential, and use the public key included thereof as asymmetric proof-of-possession key. For example, this is the case when the joining node relies on

Section 3.2.2 of [I-D.ietf-ace-dtls-authorize] and authenticates itself during the DTLS handshake with the Group Manager. However, this requires the authentication credential to be in the format used in the OSCORE group, and that both the authentication credential of the joining node and the included public key are compatible with the signature or ECDH algorithm, and possible associated parameters used in the OSCORE group.

A node may have joined multiple OSCORE groups under different non-synchronized Group Managers. Therefore, it can happen that those OSCORE groups have the same Group Identifier (Gid). It follows that, upon receiving a Group OSCORE message addressed to one of those groups, the node would have multiple Security Contexts matching with the Gid in the incoming message. It is up to the application to decide how to handle such collisions of Group Identifiers, e.g., by trying to process the incoming message using one Security Context at the time until the right one is found.

15.2. Size of Nonces as Proof-of-Possession Challenge

With reference to the Joining Request message in Section 6.1, the proof-of-possession (PoP) evidence included in 'client_cred_verify' is computed over an input including also $N_C \parallel N_S$, where \parallel denotes concatenation.

For the N_C challenge, it is RECOMMENDED to use a 8-byte long random nonce. Furthermore, N_C is always conveyed in the 'cnonce' parameter of the Joining Request, which is always sent over the secure communication association between the joining node and the Group Manager.

As defined in Section 6.1.1, the way the N_S value is computed depends on the particular way the joining node provides the Group Manager with the Access Token, as well as on following interactions between the two.

- * If the Access Token has not been provided to the Group Manager by means of a Token Transfer Request to the /authz-info endpoint as in Section 5.3, then N_S is computed as a 32-byte long challenge. For an example, see point (2) of Section 6.1.1.

- * If the Access Token has been provided to the Group Manager by means of a Token Transfer Request to the /authz-info endpoint as in Section 5.3, then N_S takes the most recent value provided to the Client by the Group Manager in the 'kdcchallenge' parameter, as specified in point (1) of Section 6.1.1. This value is provided either in the Token Transfer Response (see Section 5.3), or in a 4.00 (Bad Request) error response to a following Joining Request (see Section 6.2). In either case, it is RECOMMENDED to use a 8-byte long random challenge as value for N_S.

If we consider both N_C and N_S to take 8-byte long values, the following considerations hold.

- * Let us consider both N_C and N_S as taking random values, and the Group Manager to never change the value of the N_S provided to a Client during the lifetime of an Access Token. Then, as per the birthday paradox, the average collision for N_S will happen after 2^{32} new transferred Access Tokens, while the average collision for N_C will happen after 2^{32} new Joining Requests. This amounts to considerably more token provisionings than the expected new joinings of OSCORE groups under a same Group Manager, as well as to considerably more requests to join OSCORE groups from a same Client using a same Access Token under a same Group Manager.
- * Section 7 of [I-D.ietf-ace-oscore-profile] as well Appendix B.2 of [RFC8613] recommend the use of 8-byte random values as well. Unlike in those cases, the values of N_C and N_S considered in this document are not used for as sensitive operations as the derivation of a Security Context, and thus do not have possible implications in the security of AEAD ciphers.

15.3. Reusage of Nonces for Proof-of-Possession Input

As long as the Group Manager preserves the same N_S value currently associated with an Access Token, i.e., the latest value provided to a Client in a 'kdcchallenge' parameter, the Client is able to successfully reuse the same proof-of-possession (PoP) input for multiple Joining Requests to that Group Manager.

In particular, the Client can reuse the same N_C value for every Joining Request to the Group Manager, and combine it with the same unchanged N_S value. This results in reusing the same PoP input for producing the PoP evidence to include in the 'client_cred_verify' parameter of the Joining Requests.

Unless the Group Manager maintains a list of N_C values already used by that Client since the latest update to the N_S value associated with the Access Token, the Group Manager can be forced to falsely

believe that the Client possesses its own private key at that point in time, upon verifying the PoP evidence in the 'client_cred_verify' parameter.

16. IANA Considerations

Note to RFC Editor: Please replace all occurrences of "[[This document]]" with the RFC number of this specification and delete this paragraph.

This document has the following actions for IANA.

16.1. OAuth Parameters

IANA is asked to register the following entries to the "OAuth Parameters" registry, as per the procedure specified in Section 11.2 of [RFC6749].

- * Parameter name: ecdh_info
- * Parameter usage location: client-rs request, rs-client response
- * Change Controller: IESG
- * Specification Document(s): [[This document]]

- * Parameter name: kdc_dh_creds
- * Parameter usage location: client-rs request, rs-client response
- * Change Controller: IESG
- * Specification Document(s): [[This document]]

16.2. OAuth Parameters CBOR Mappings

IANA is asked to register the following entries to the "OAuth Parameters CBOR Mappings" registry, as per the procedure specified in Section 8.10 of [I-D.ietf-ace-oauth-authz].

- * Name: ecdh_info
- * CBOR Key: TBD (range -256 to 255)
- * Value Type: Simple value "null" / Array
- * Reference: [[This document]]

- * Name: kdc_dh_creds
- * CBOR Key: TBD (range -256 to 255)
- * Value Type: Simple value "null" / Array
- * Reference: [[This document]]

16.3. ACE Groupcomm Parameters

IANA is asked to register the following entry to the "ACE Groupcomm Parameters" registry defined in Section 11.7 of [I-D.ietf-ace-key-groupcomm].

- * Name: group_senderId
 - * CBOR Key: TBD
 - * CBOR Type: Byte string
 - * Reference: [[This document]] (Section 9.2)
-
- * Name: ecdh_info
 - * CBOR Key: TBD
 - * CBOR Type: Array
 - * Reference: [[This document]] (Section 6.2)
-
- * Name: kdc_dh_creds
 - * CBOR Key: TBD
 - * CBOR Type: Array
 - * Reference: [[This document]] (Section 6.2)
-
- * Name: group_enc_key
 - * CBOR Key: TBD
 - * CBOR Type: Byte string
 - * Reference: [[This document]] (Section 8.2.1)

- * Name: stale_node_ids
- * CBOR Key: TBD
- * CBOR Type: Array
- * Reference: [[This document]] (Section 11)

16.4. ACE Groupcomm Key Types

IANA is asked to register the following entry to the "ACE Groupcomm Key Types" registry defined in Section 11.8 of [I-D.ietf-ace-key-groupcomm].

- * Name: Group_OSCORE_Input_Material object
- * Key Type Value: GROUPCOMM_KEY_TBD
- * Profile: "coap_group_oscore_app", defined in Section 16.5 of this document.
- * Description: A Group_OSCORE_Input_Material object encoded as described in Section 6.3 of this document.
- * Reference: [[This document]] (Section 6.3)

16.5. ACE Groupcomm Profiles

IANA is asked to register the following entry to the "ACE Groupcomm Profiles" registry defined in Section 11.9 of [I-D.ietf-ace-key-groupcomm].

- * Name: coap_group_oscore_app
- * Description: Application profile to provision keying material for participating in group communication protected with Group OSCORE as per [I-D.ietf-core-oscore-groupcomm].
- * CBOR Value: PROFILE_TBD
- * Reference: [[This document]] (Section 6.3)

16.6. OSCORE Security Context Parameters

IANA is asked to register the following entries in the "OSCORE Security Context Parameters" registry defined in Section 9.4 of [I-D.ietf-ace-oscore-profile].

- * Name: group_SenderId
 - * CBOR Label: TBD
 - * CBOR Type: Byte string
 - * Registry: -
 - * Description: OSCORE Sender ID assigned to a member of an OSCORE group
 - * Reference: [[This document]] (Section 6.3)
-
- * Name: cred_fmt
 - * CBOR Label: TBD
 - * CBOR Type: Integer
 - * Registry: COSE Header Parameters
 - * Description: Format of authentication credentials to be used in the OSCORE group
 - * Reference: [[This document]] (Section 6.3)
-
- * Name: sign_enc_alg
 - * CBOR Label: TBD
 - * CBOR Type: Text string / Integer
 - * Registry: COSE Algorithms
 - * Description: OSCORE Signature Encryption Algorithm Value
 - * Reference: [[This document]] (Section 6.3)
-
- * Name: sign_alg
 - * CBOR Label: TBD
 - * CBOR Type: Text string / Integer
 - * Registry: COSE Algorithms

- * Description: OSCORE Signature Algorithm Value
- * Reference: [[This document]] (Section 6.3)

- * Name: sign_params
- * CBOR Label: TBD
- * CBOR Type: Array
- * Registry: COSE Algorithms, COSE Key Types, COSE Elliptic Curves
- * Description: OSCORE Signature Algorithm Parameters
- * Reference: [[This document]] (Section 6.3)

- * Name: ecdh_alg
- * CBOR Label: TBD
- * CBOR Type: Text string / Integer
- * Registry: COSE Algorithms
- * Description: OSCORE Pairwise Key Agreement Algorithm Value
- * Reference: [[This document]] (Section 6.3)

- * Name: ecdh_params
- * CBOR Label: TBD
- * CBOR Type: Array
- * Registry: COSE Algorithms, COSE Key Types, COSE Elliptic Curves
- * Description: OSCORE Pairwise Key Agreement Algorithm Parameters
- * Reference: [[This document]] (Section 6.3)

16.7. TLS Exporter Labels

IANA is asked to register the following entry to the "TLS Exporter Labels" registry defined in Section 6 of [RFC5705] and updated in Section 12 of [RFC8447].

- * Value: EXPORTER-ACE-Sign-Challenge-coap-group-oscore-app
- * DTLS-OK: Y
- * Recommended: N
- * Reference: [[This document]] (Section 6.1.1)

16.8. AIF

For the media-types `application/aif+cbor` and `application/aif+json` defined in Section 5.1 of [I-D.ietf-ace-aif], IANA is requested to register the following entries for the two media-type parameters `Toid` and `Tperm`, in the respective sub-registry defined in Section 5.2 of [I-D.ietf-ace-aif] within the "MIME Media Type Sub-Parameter" registry group.

- * Name: `oscore-gname`
- * Description/Specification: OSCORE group name
- * Reference: [[This document]]
- * Name: `oscore-gperm`
- * Description/Specification: permissions pertaining OSCORE groups
- * Reference: [[This document]]

16.9. CoAP Content-Format

IANA is asked to register the following entries to the "CoAP Content-Formats" registry within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

- * Media Type: `application/aif+cbor;Toid="oscore-gname",Tperm="oscore-gperm"`
- * Encoding: -
- * ID: TBD
- * Reference: [[This document]]
- * Media Type: `application/aif+json;Toid="oscore-gname",Tperm="oscore-gperm"`

- * Encoding: -
- * ID: TBD
- * Reference: [[This document]]

16.10. Group OSCORE Roles

This document establishes the IANA "Group OSCORE Roles" registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 16.14.

This registry includes the possible roles that nodes can take in an OSCORE group, each in combination with a numeric identifier. These numeric identifiers are used to express authorization information about joining OSCORE groups, as specified in Section 3 of [[This document]].

The columns of this registry are:

- * Name: A value that can be used in documents for easier comprehension, to identify a possible role that nodes can take in an OSCORE group.
- * Value: The numeric identifier for this role. Integer values greater than 65535 are marked as "Private Use", all other values use the registration policy "Expert Review" [RFC8126].
- * Description: This field contains a brief description of the role.
- * Reference: This contains a pointer to the public specification for the role.

This registry will be initially populated by the values in Figure 1.

The Reference column for all of these entries will be [[This document]].

16.11. CoRE Resource Type

IANA is asked to register the following entry in the "Resource Type (rt=) Link Target Attribute Values" registry within the "Constrained Restful Environments (CoRE) Parameters" registry group.

- * Value: "core.osc.gm"
- * Description: Group-membership resource of an OSCORE Group Manager.

- * Reference: [[This document]]

Client applications can use this resource type to discover a group membership resource at an OSCORE Group Manager, where to send a request for joining the corresponding OSCORE group.

16.12. ACE Scope Semantics

IANA is asked to register the following entry in the "ACE Scope Semantics" registry defined in Section 11.12 of [I-D.ietf-ace-key-groupcomm].

- * Value: SEM_ID_TBD
- * Description: Membership and key management operations at the ACE Group Manager for Group OSCORE.
- * Reference: [[This document]]

16.13. ACE Groupcomm Errors

IANA is asked to register the following entry in the "ACE Groupcomm Errors" registry defined in Section 11.13 of [I-D.ietf-ace-key-groupcomm].

- * Value: 7
- * Description: Signatures not used in the group.
- * Reference: [[This document]]
- * Value: 8
- * Description: Operation permitted only to signature verifiers.
- * Reference: [[This document]]
- * Value: 9
- * Description: Group currently not active.
- * Reference: [[This document]]

16.14. Expert Review Instructions

The IANA registry established in this document is defined as "Expert Review". This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

- * Clarity and correctness of registrations. Experts are expected to check the clarity of purpose and use of the requested entries. Experts should inspect the entry for the considered role, to verify the correctness of its description against the role as intended in the specification that defined it. Experts should consider requesting an opinion on the correctness of registered parameters from the Authentication and Authorization for Constrained Environments (ACE) Working Group and the Constrained RESTful Environments (CoRE) Working Group.

Entries that do not meet these objective of clarity and completeness should not be registered.

- * Duplicated registration and point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments.
- * Experts should take into account the expected usage of roles when approving point assignment. Given a 'Value' V as code point, the length of the encoding of $(2^{(V+1)} - 1)$ should be weighed against the usage of the entry, considering the resources and capabilities of devices it will be used on. Additionally, given a 'Value' V as code point, the length of the encoding of $(2^{(V+1)} - 1)$ should be weighed against how many code points resulting in that encoding length are left, and the resources and capabilities of devices it will be used on.
- * Specifications are recommended. When specifications are not provided, the description provided needs to have sufficient information to verify the points above.

17. References

17.1. Normative References

[COSE.Algorithms]
IANA, "COSE Algorithms",
<<https://www.iana.org/assignments/cose/cose.xhtml#algorithms>>.

[COSE.Elliptic.Curves]
IANA, "COSE Elliptic Curves",
<<https://www.iana.org/assignments/cose/cose.xhtml#elliptic-curves>>.

[COSE.Header.Parameters]
IANA, "COSE Header Parameters",
<<https://www.iana.org/assignments/cose/cose.xhtml#header-parameters>>.

[COSE.Key.Types]
IANA, "COSE Key Types",
<<https://www.iana.org/assignments/cose/cose.xhtml#key-type>>.

[I-D.ietf-ace-aif]
Bormann, C., "An Authorization Information Format (AIF) for ACE", Work in Progress, Internet-Draft, draft-ietf-ace-aif-07, 15 March 2022,
<<https://www.ietf.org/archive/id/draft-ietf-ace-aif-07.txt>>.

[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)", Work in Progress, Internet-Draft, draft-ietf-ace-dtls-authorize-18, 4 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-ace-dtls-authorize-18.txt>>.

[I-D.ietf-ace-key-groupcomm]
Palombini, F. and M. Tiloca, "Key Provisioning for Group Communication using ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-15, 23 December 2021, <<https://www.ietf.org/archive/id/draft-ietf-ace-key-groupcomm-15.txt>>.

[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)", Work in Progress, Internet-Draft,

draft-ietf-ace-oauth-authz-46, 8 November 2021,
<<https://www.ietf.org/archive/id/draft-ietf-ace-oauth-authz-46.txt>>.

[I-D.ietf-ace-oscore-profile]

Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson,
"OSCORE Profile of the Authentication and Authorization
for Constrained Environments Framework", Work in Progress,
Internet-Draft, draft-ietf-ace-oscore-profile-19, 6 May
2021, <<https://www.ietf.org/archive/id/draft-ietf-ace-oscore-profile-19.txt>>.

[I-D.ietf-core-oscore-groupcomm]

Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P.,
and J. Park, "Group OSCORE - Secure Group Communication
for CoAP", Work in Progress, Internet-Draft, draft-ietf-
core-oscore-groupcomm-14, 7 March 2022,
<<https://www.ietf.org/archive/id/draft-ietf-core-oscore-groupcomm-14.txt>>.

[I-D.ietf-cose-rfc8152bis-algs]

Schaad, J., "CBOR Object Signing and Encryption (COSE):
Initial Algorithms", Work in Progress, Internet-Draft,
draft-ietf-cose-rfc8152bis-algs-12, 24 September 2020,
<<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-algs-12.txt>>.

[I-D.ietf-cose-rfc8152bis-struct]

Schaad, J., "CBOR Object Signing and Encryption (COSE):
Structures and Process", Work in Progress, Internet-Draft,
draft-ietf-cose-rfc8152bis-struct-15, 1 February 2021,
<<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-struct-15.txt>>.

[NIST-800-56A]

Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R.
Davis, "Recommendation for Pair-Wise Key-Establishment
Schemes Using Discrete Logarithm Cryptography - NIST
Special Publication 800-56A, Revision 3", April 2018,
<<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf>>.

[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>>.

- [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>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.
- [RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <<https://www.rfc-editor.org/info/rfc6979>>.
- [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>>.
- [RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <<https://www.rfc-editor.org/info/rfc7748>>.
- [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <<https://www.rfc-editor.org/info/rfc8017>>.
- [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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.
- [RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <<https://www.rfc-editor.org/info/rfc8613>>.
- [RFC8742] Bormann, C., "Concise Binary Object Representation (CBOR) Sequences", RFC 8742, DOI 10.17487/RFC8742, February 2020, <<https://www.rfc-editor.org/info/rfc8742>>.
- [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>>.

17.2. Informative References

- [I-D.ietf-ace-oscore-gm-admin]
Tiloca, M., Höglund, R., Stok, P. V. D., and F. Palombini, "Admin Interface for the OSCORE Group Manager", Work in Progress, Internet-Draft, draft-ietf-ace-oscore-gm-admin-05, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-ace-oscore-gm-admin-05.txt>>.
- [I-D.ietf-core-coap-pubsub]
Koster, M., Keranen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-coap-pubsub-09, 30 September 2019, <<https://www.ietf.org/archive/id/draft-ietf-core-coap-pubsub-09.txt>>.
- [I-D.ietf-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-06, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-core-groupcomm-bis-06.txt>>.

- [I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-03, 10 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-cose-cbor-encoded-cert-03.txt>>.
- [I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsuess, C., and P. V. D. Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-discovery-11, 7 March 2022, <<https://www.ietf.org/archive/id/draft-tiloca-core-oscore-discovery-11.txt>>.
- [RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <<https://www.rfc-editor.org/info/rfc6690>>.
- [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>>.
- [RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <<https://www.rfc-editor.org/info/rfc7641>>.
- [RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <<https://www.rfc-editor.org/info/rfc7925>>.
- [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>>.

Appendix A. Profile Requirements

This section lists how this application profile of ACE addresses the requirements defined in Appendix A of [I-D.ietf-ace-key-groupcomm].

A.1. Mandatory-to-Address Requirements

- * REQ1 - Specify the format and encoding of 'scope'. This includes defining the set of possible roles and their identifiers, as well as the corresponding encoding to use in the scope entries according to the used scope format: see Section 3 and Section 5.1.
- * REQ2 - If the AIF format of 'scope' is used, register its specific instance of "Toid" and "Tperm" as Media Type parameters and a corresponding Content-Format, as per the guidelines in [I-D.ietf-ace-aif]: see Section 16.8 and Section 16.9.
- * REQ3 - if used, specify the acceptable values for 'sign_alg': values from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].
- * REQ4 - If used, specify the acceptable values for 'sign_parameters': format and values from the COSE algorithm capabilities as specified in the "COSE Algorithms" registry [COSE.Algorithms].
- * REQ5 - If used, specify the acceptable values for 'sign_key_parameters': format and values from the COSE key type capabilities as specified in the "COSE Key Types" registry [COSE.Key.Types].
- * REQ6 - Specify the acceptable formats for authentication credentials and, if used, the acceptable values for 'pub_key_enc': acceptable formats explicitly provide the public key as well as the comprehensive set of information related to the public key algorithm (see Section 5.3 and Section 6.3). Consistent acceptable values for 'pub_key_enc' are taken from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters].
- * REQ7 - If the value of the GROUPNAME URI path and the group name in the Access Token scope (gname in Section 3.1 of [I-D.ietf-ace-key-groupcomm]) are not required to coincide, specify the mechanism to map the GROUPNAME value in the URI to the group name: not applicable, since a perfect matching is required.

- * REQ8 - Define whether the KDC has an authentication credential and if this has to be provided through the 'kdc_cred' parameter, see Section 4.1 of [I-D.ietf-ace-key-groupcomm]: yes, as required by the Group OSCORE protocol [I-D.ietf-core-oscore-groupcomm], see Section 6.3 of this document.
- * REQ9 - Specify if any part of the KDC interface as defined in Section 4.1 of [I-D.ietf-ace-key-groupcomm] is not supported by the KDC: not applicable.
- * REQ10 - Register a Resource Type for the root url-path, which is used to discover the correct url to access at the KDC (see Section 4.1 of [I-D.ietf-ace-key-groupcomm]): the Resource Type (rt=) Link Target Attribute value "core.osc.gm" is registered in Section 16.11.
- * REQ11 - Define what specific actions (e.g., CoAP methods) are allowed on each resource provided by the KDC interface, depending on whether the Client is a current group member; the roles that a Client is authorized to take as per the obtained access token; and the roles that the Client has as current group member: see Section 8.4.
- * REQ12 - Categorize possible newly defined operations for Clients into primary operations expected to be minimally supported and secondary operations, and provide accompanying considerations: see Section 8.5.
- * REQ13 - Specify the encoding of group identifier (see Section 4.2.1 of [I-D.ietf-ace-key-groupcomm]): CBOR byte string (see Section 9.10).
- * REQ14 - Specify the approaches used to compute and verify the PoP evidence to include in 'client_cred_verify', and which of those approaches is used in which case: see Section 6.1 and Section 6.2.
- * REQ15 - Specify how the nonce N_S is generated, if the token is not provided to the KDC through the Token Transfer Request to the authz-info endpoint (e.g., if it is used directly to validate TLS instead): see Section 6.1.1.
- * REQ16 - Define the initial value of the 'num' parameter: the initial value MUST be set to 0 when creating the OSCORE group, e.g., as in [I-D.ietf-ace-oscore-gm-admin].
- * REQ17 - Specify the format of the 'key' parameter: see Section 6.3.

- * REQ18 - Specify acceptable values of the 'gkty' parameter: Group_OSCORE_Input_Material object (see Section 6.3).
- * REQ19 - Specify and register the application profile identifier: coap_group_oscore_app (see Section 16.5).
- * REQ20 - If used, specify the format and content of 'group_policies' and its entries: see Section 6.3.
- * REQ21 - Specify the approaches used to compute and verify the PoP evidence to include in 'kdc_cred_verify', and which of those approaches is used in which case: see Section 6.3, Section 6.4 and Section 9.5.
- * REQ22 - Specify the communication protocol that the members of the group must use: CoAP [RFC7252], also for group communication [I-D.ietf-core-groupcomm-bis].
- * REQ23 - Specify the security protocols that the group members must use to protect their communication: Group OSCORE [I-D.ietf-core-oscore-groupcomm].
- * REQ24 - Specify how the communication is secured between the Client and KDC: by means of any transport profile of ACE [I-D.ietf-ace-oauth-authz] between Client and Group Manager that complies with the requirements in Appendix C of [I-D.ietf-ace-oauth-authz].
- * REQ25 - Specify the format of the identifiers of group members: the Sender ID used in the OSCORE group (see Section 6.3 and Section 9.3).
- * REQ26 - Specify policies at the KDC to handle member ids that are not included in 'get_pub_keys': see Section 9.3.
- * REQ27 - Specify the format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label: see Section 9.2.
- * REQ28 - Specify and register the identifier of newly defined semantics for binary scopes: see Section 16.12.
- * REQ29 - Categorize newly defined parameters according to the same criteria of Section 8 of [I-D.ietf-ace-key-groupcomm]: see Section 12.

- * REQ30 - Define whether Clients must, should or may support the conditional parameters defined in Section 8 of [I-D.ietf-ace-key-groupcomm], and under which circumstances: see Section 12.

A.2. Optional-to-Address Requirements

- * OPT1 (Optional) - If the textual format of 'scope' is used, specify CBOR values to use for abbreviating the role identifiers in the group: not applicable.
- * OPT2 (Optional) - Specify additional parameters used in the exchange of Token Transfer Request and Response:
 - 'ecdh_info', to negotiate the ECDH algorithm, ECDH algorithm parameters, ECDH key parameters and exact format of authentication credentials used in the group, in case the joining node supports the pairwise mode of Group OSCORE (see Section 5.3).
 - 'kdc_dh_creds', to ask for and retrieve the Group Manager's Diffie-Hellman authentication credentials, in case the joining node supports the pairwise mode of Group OSCORE and the Access Token authorizes to join pairwise-only groups (see Section 5.3).
- * OPT3 (Optional) - Specify the negotiation of parameter values for signature algorithm and signature keys, if 'sign_info' is not used: possible early discovery by using the approach based on the CoRE Resource Directory described in [I-D.tiloca-core-oscore-discovery].
- * OPT4 (Optional) - Specify possible or required payload formats for specific error cases: send a 4.00 (Bad Request) error response to a Joining Request (see Section 6.2).
- * OPT5 (Optional) - Specify additional identifiers of error types, as values of the 'error' field in an error response from the KDC: see Section 16.13.
- * OPT6 (Optional) - Specify the encoding of 'pub_keys_repos' if the default is not used: no.
- * OPT7 (Optional) - Specify the functionalities implemented at the 'control_uri' resource hosted at the Client, including message exchange encoding and other details (see Section 4.3.1 of [I-D.ietf-ace-key-groupcomm]): see Section 10 for the eviction of a group member; see Section 11 for the group rekeying process.

- * OPT8 (Optional) - Specify the behavior of the handler in case of failure to retrieve an authentication credential for the specific node: send a 4.00 (Bad Request) error response to a Joining Request (see Section 6.2).
- * OPT9 (Optional) - Define a default group rekeying scheme, to refer to in case the 'rekeying_scheme' parameter is not included in the Joining Response (see Section 4.3.1.1 of [I-D.ietf-ace-key-groupcomm]): the "Point-to-Point" rekeying scheme registered in Section 11.14 of [I-D.ietf-ace-key-groupcomm], whose detailed use for this profile is defined in Section 11 of this document.
- * OPT10 (Optional) - Specify the functionalities implemented at the 'control_group_uri' resource hosted at the Client, including message exchange encoding and other details (see Section 4.3.1 of [I-D.ietf-ace-key-groupcomm]): see Section 10 for the eviction of multiple group members.
- * OPT11 (Optional) - Specify policies that instruct Clients to retain unsuccessfully decrypted messages and for how long, so that they can be decrypted after getting updated keying material: no.
- * OPT12 (Optional) - Specify for the KDC to perform group rekeying (together or instead of renewing individual keying material) when receiving a Key Renewal Request: the Group Manager SHOULD NOT perform a group rekeying, unless already scheduled to occur shortly (see Section 9.2).
- * OPT13 (Optional) - Specify how the identifier of a group members's authentication credential is included in requests sent to other group members: no.
- * OPT14 (Optional) - Specify additional information to include in rekeying messages for the "Point-to-Point" group rekeying scheme (see Section 6.1 of [I-D.ietf-ace-key-groupcomm]): see Section 11.1.
- * OPT15 (Optional) - Specify if Clients must or should support any of the parameters defined as optional in Section 8 of [I-D.ietf-ace-key-groupcomm]: no.

Appendix B. Extensibility for Future COSE Algorithms

As defined in Section 8.1 of [I-D.ietf-cose-rfc8152bis-algs], future algorithms can be registered in the "COSE Algorithms" registry [COSE.Algorithms] as specifying none or multiple COSE capabilities.

To enable the seamless use of such future registered algorithms, this section defines a general, agile format for:

- * Each 'ecdh_info_entry' of the 'ecdh_info' parameter in the Token Transfer Response (see Section 5.3 and Section 5.3.1);
- * The 'sign_params' and 'ecdh_params' parameters within the 'key' parameter (see Section 6.3), as part of the response payloads used in Section 6.3, Section 9.1.1, Section 9.1.2 and Section 11.

Appendix B of [I-D.ietf-ace-key-groupcomm] describes the analogous general format for 'sign_info_entry' of the 'sign_info' parameter in the Token Transfer Response (see Section 5.3 of this document).

If any of the currently registered COSE algorithms is considered, using this general format yields the same structure defined in this document for the items above, thus ensuring retro-compatibility.

B.1. Format of 'ecdh_info_entry'

The format of each 'ecdh_info_entry' (see Section 5.3 and Section 5.3.1) is generalized as follows. Given N the number of elements of the 'ecdh_parameters' array, i.e., the number of COSE capabilities of the ECDH algorithm, then:

- * 'ecdh_key_parameters' is replaced by N elements 'ecdh_capab_i', each of which is a CBOR array.
- * The i-th array following 'ecdh_parameters', i.e., 'ecdh_capab_i' (i = 0, ..., N-1), is the array of COSE capabilities for the algorithm capability specified in 'ecdh_parameters'[i].

```
ecdh_info_entry =  
[  
  id : gname / [ + gname ],  
  ecdh_alg : int / tstr,  
  ecdh_parameters : [ alg_capab_1 : any,  
                     alg_capab_2 : any,  
                     ...,  
                     alg_capab_N : any ],  
  ecdh_capab_1 : [ any ],  
  ecdh_capab_2 : [ any ],  
  ...,  
  ecdh_capab_N : [ any ],  
  cred_fmt = int / null  
]  
  
gname = tstr
```

Figure 13: 'ecdh_info_entry' with general format

B.2. Format of 'key'

The format of 'key' (see Section 6.3) is generalized as follows.

- * The 'sign_params' array includes N+1 elements, whose exact structure and value depend on the value of the signature algorithm specified in 'sign_alg'.
 - The first element, i.e., 'sign_params'[0], is the array of the N COSE capabilities for the signature algorithm, as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms] (see Section 8.1 of [I-D.ietf-cose-rfc8152bis-algs]).
 - Each following element 'sign_params'[i], i.e., with index i > 0, is the array of COSE capabilities for the algorithm capability specified in 'sign_params'[0][i-1].

For example, if 'sign_params'[0][0] specifies the key type as capability of the algorithm, then 'sign_params'[1] is the array of COSE capabilities for the COSE key type associated with the signature algorithm, as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types] (see Section 8.2 of [I-D.ietf-cose-rfc8152bis-algs]).

- * The 'ecdh_params' array includes M+1 elements, whose exact structure and value depend on the value of the ECDH algorithm specified in 'ecdh_alg'.
 - The first element, i.e., 'ecdh_params'[0], is the array of the M COSE capabilities for the ECDH algorithm, as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms] (see Section 8.1 of [I-D.ietf-cose-rfc8152bis-algs]).
 - Each following element 'ecdh_params'[i], i.e., with index i > 0, is the array of COSE capabilities for the algorithm capability specified in 'ecdh_params'[0][i-1].

For example, if 'ecdh_params'[0][0] specifies the key type as capability of the algorithm, then 'ecdh_params'[1] is the array of COSE capabilities for the COSE key type associated with the ECDH algorithm, as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types] (see Section 8.2 of [I-D.ietf-cose-rfc8152bis-algs]).

Appendix C. Document Updates

RFC EDITOR: PLEASE REMOVE THIS SECTION.

C.1. Version -13 to -14

- * Major reordering of the document sections.
- * The HKDF Algorithm is specified by the HMAC Algorithm.
- * Group communication does not necessarily use IP multicast.
- * Generalized AIF data model, also for draft-ace-oscore-gm-admin.
- * Clarifications and editorial improvements.

C.2. Version -12 to -13

- * Renamed parameters about authentication credentials.
- * It is optional for the Group Manager to reassign Gids by tracking "Birth Gids".
- * Distinction between authentication credentials and public keys.
- * Updated IANA considerations related to AIF.
- * Updated textual description of registered ACE Scope Semantics value.

C.3. Version -11 to -12

- * Clarified semantics of 'ecdh_info' and 'kdc_dh_creds'.
- * Definition of /ace-group/GROUPNAME/kdc-pub-key moved to draft-ietf-ace-key-groupcomm.
- * ace-group/ accessible also to non-members that are not Verifiers.
- * Clarified what resources are accessible to Verifiers.
- * Revised error handling for the newly defined resources.
- * Revised use of CoAP error codes.
- * Use of "Token Tranfer Request" and "Token Transfer Response".
- * New parameter 'rekeying_scheme'.

- * Categorization of new parameters and inherited conditional parameters.
- * Clarifications on what to do in case of enhanced error responses.
- * Changed UCCS to CCS.
- * Authentication credentials of just joined Clients can be in rekeying messages.
- * Revised names of new IANA registries.
- * Clarified meaning of registered CoRE resource type.
- * Alignment to new requirements from draft-ietf-ace-key-groupcomm.
- * Fixes and editorial improvements.

C.4. Version -10 to -11

- * Removed redundancy of key type capabilities, from 'sign_info', 'ecdh_info' and 'key'.
- * New resource to retrieve the Group Manager's authentication credential.
- * New resource to retrieve material for Signature Verifiers.
- * New parameter 'sign_enc_alg' related to the group mode.
- * 'cred_fmt' takes value from the COSE Header Parameters registry.
- * Improved alignment of the Joining Response payload with the Group OSCORE Security Context parameters.
- * Recycling Group IDs by tracking "Birth GIDs".
- * Error handling in case of non available Sender IDs upon joining.
- * Error handling in case EdDSA public keys with invalid Y coordinate when the pairwise mode of Group OSCORE is supported.
- * Generalized proof-of-possession (PoP) for the joining node's private key; defined Diffie-Hellman based PoP for OSCORE groups using only the pairwise mode.
- * Proof-of-possession of the Group Manager's private key in the Joining Response.

- * Always use 'peer_identifiers' to convey Sender IDs as node identifiers.
- * Stale Sender IDs provided when rekeying the group.
- * New resource for late retrieval of stale Sender IDs.
- * Added examples of message exchanges.
- * Revised default values of group configuration parameters.
- * Fixes to IANA registrations.
- * General format of parameters related to COSE capabilities, supporting future registered COSE algorithms (new Appendix).

C.5. Version -09 to -10

- * Updated non-recycling policy of Sender IDs.
- * Removed policies about Sender Sequence Number synchronization.
- * 'control_path' renamed to 'control_uri'.
- * Format of 'get_pub_keys' aligned with draft-ietf-ace-key-groupcomm.
- * Additional way to inform of group eviction.
- * Registered semantics identifier for extended scope format.
- * Extended error handling, with error type specified in some error responses.
- * Renumbered requirements.

C.6. Version -08 to -09

- * The url-path "ace-group" is used.
- * Added overview of admitted methods on the Group Manager resources.
- * Added exchange of parameters relevant for the pairwise mode of Group OSCORE.
- * The signed value for 'client_cred_verify' includes also the scope.

- * Renamed the key material object as Group_OSCORE_Input_Material object.
- * Replaced 'clientId' with 'group_SenderId'.
- * Added message exchange for Group Names request-response.
- * No reassignment of Sender ID and Gid in the same OSCORE group.
- * Updates on group rekeying contextual with request of new Sender ID.
- * Signature verifiers can also retrieve Group Names and URIs.
- * Removed group policy about supporting Group OSCORE in pairwise mode.
- * Registration of the resource type rt="core.osc.gm".
- * Update list of requirements.
- * Clarifications and editorial revision.

C.7. Version -07 to -08

- * AIF specific data model to express scope entries.
- * A set of roles is checked as valid when processing the Joining Request.
- * Updated format of 'get_pub_keys' in the Joining Request.
- * Payload format and default values of group policies in the Joining Response.
- * Updated payload format of the FETCH request to retrieve authentication credentials.
- * Default values for group configuration parameters.
- * IANA registrations to support the AIF specific data model.

C.8. Version -06 to -07

- * Alignments with draft-ietf-core-oscore-groupcomm.
- * New format of 'sign_info', using the COSE capabilities.

- * New format of Joining Response parameters, using the COSE capabilities.
- * Considerations on group rekeying.
- * Editorial revision.

C.9. Version -05 to -06

- * Added role of external signature verifier.
- * Parameter 'rsnonce' renamed to 'kdcchallenge'.
- * Parameter 'kdcchallenge' may be omitted in some cases.
- * Clarified difference between group name and OSCORE Gid.
- * Removed the role combination ["requester", "monitor"].
- * Admit implicit scope and audience in the Authorization Request.
- * New format for the 'sign_info' parameter.
- * Scope not mandatory to include in the Joining Request.
- * Group policy about supporting Group OSCORE in pairwise mode.
- * Possible individual rekeying of a single requesting node combined with a group rekeying.
- * Security considerations on reuse of signature challenges.
- * Addressing optional requirement OPT12 from draft-ietf-ace-key-groupcomm
- * Editorial improvements.

C.10. Version -04 to -05

- * Nonce N_S also in error responses to the Joining Requests.
- * Supporting single Access Token for multiple groups/topics.
- * Supporting legal requesters/responders using the 'peer_roles' parameter.
- * Registered and used dedicated label for TLS Exporter.

- * Added method for uploading a new authentication credential to the Group Manager.
- * Added resource and method for retrieving the current group status.
- * Fixed inconsistency in retrieving group keying material only.
- * Clarified retrieval of keying material for monitor-only members.
- * Clarification on incrementing version number when rekeying the group.
- * Clarification on what is re-distributed with the group rekeying.
- * Security considerations on the size of the nonces used for the signature challenge.
- * Added CBOR values to abbreviate role identifiers in the group.

C.11. Version -03 to -04

- * New abstract.
- * Moved general content to draft-ietf-ace-key-groupcomm
- * Terminology: node name; node resource.
- * Creation and pointing at node resource.
- * Updated Group Manager API (REST methods and offered services).
- * Size of challenges 'cnonce' and 'rsnonce'.
- * Value of 'rsnonce' for reused or non-traditionally-posted tokens.
- * Removed reference to RFC 7390.
- * New requirements from draft-ietf-ace-key-groupcomm
- * Editorial improvements.

C.12. Version -02 to -03

- * New sections, aligned with the interface of ace-key-groupcomm .
- * Exchange of information on the signature algorithm and related parameters, during the Token POST (Section 4.1).

- * Nonce 'rsnonce' from the Group Manager to the Client (Section 4.1).
- * Client PoP signature in the Key Distribution Request upon joining (Section 4.2).
- * Local actions on the Group Manager, upon a new node's joining (Section 4.2).
- * Local actions on the Group Manager, upon a node's leaving (Section 12).
- * IANA registration in ACE Groupcomm Parameters registry.
- * More fulfilled profile requirements (Appendix A).

C.13. Version -01 to -02

- * Editorial fixes.
- * Changed: "listener" to "responder"; "pure listener" to "monitor".
- * Changed profile name to "coap_group_oscore_app", to reflect it is an application profile.
- * Added the 'type' parameter for all requests to a Join Resource.
- * Added parameters to indicate the encoding of authentication credentials.
- * Challenge-response for proof-of-possession of signature keys (Section 4).
- * Renamed 'key_info' parameter to 'sign_info'; updated its format; extended to include also parameters of the signature key (Section 4.1).
- * Code 4.00 (Bad request), in responses to joining nodes providing an invalid authentication credential (Section 4.3).
- * Clarifications on provisioning and checking of authentication credentials (Sections 4 and 6).
- * Extended discussion on group rekeying and possible different approaches (Section 7).
- * Extended security considerations: proof-of-possession of signature keys; collision of OSCORE Group Identifiers (Section 8).

- * Registered three entries in the IANA registry "Sequence Number Synchronization Method" (Section 9).
- * Registered one public key encoding in the "ACE Public Key Encoding" IANA registry (Section 9).

C.14. Version -00 to -01

- * Changed name of 'req_aud' to 'audience' in the Authorization Request (Section 3.1).
- * Added negotiation of signature algorithm/parameters between Client and Group Manager (Section 4).
- * Updated format of the Key Distribution Response as a whole (Section 4.3).
- * Added parameter 'cs_params' in the 'key' parameter of the Key Distribution Response (Section 4.3).
- * New IANA registrations in the "ACE Authorization Server Request Creation Hints" registry, "ACE Groupcomm Key" registry, "OSCORE Security Context Parameters" registry and "ACE Groupcomm Profile" registry (Section 9).

Acknowledgments

The authors sincerely thank Christian Amsuess, Santiago Aragon, Stefan Beck, Carsten Bormann, Martin Gunnarsson, Rikard Hoeglund, Watson Ladd, Daniel Migault, Jim Schaad, Ludwig Seitz, Goeran Selander and Peter van der Stok for their comments and feedback.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; by the H2020 project SIFIS-Home (Grant agreement 952652); and by the EIT-Digital High Impact Initiative ACTIVE.

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-164 29 Stockholm Kista
Sweden
Email: marco.tiloca@ri.se

Jiye Park
Universitaet Duisburg-Essen
Schuetzenbahn 70
45127 Essen
Germany
Email: ji-ye.park@uni-due.de

Francesca Palombini
Ericsson AB
Torshamnsgatan 23
SE-16440 Stockholm Kista
Sweden
Email: francesca.palombini@ericsson.com

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 24 September 2022

C.S. Sengul
Brunel University
A.K. Kirby
Oxbotica
23 March 2022

Message Queuing Telemetry Transport (MQTT)-TLS profile of Authentication
and Authorization for Constrained Environments (ACE) Framework
draft-ietf-ace-mqtt-tls-profile-17

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 24 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
1.1. Requirements Language	4
1.2. ACE-Related Terminology	4
1.3. MQTT-Related Terminology	5
2. Authorizing Connection Requests	9
2.1. Client Token Request to the Authorization Server (AS) . .	10
2.2. Client Connection Request to the Broker (C)	11
2.2.1. Overview of Client-RS Authentication Methods over TLS and MQTT	12
2.2.2. authz-info: The Authorization Information Topic . . .	13
2.2.3. Client Authentication over TLS	14
2.2.3.1. Raw Public Key Mode	14
2.2.3.2. Pre-Shared Key Mode	15
2.2.4. Client Authentication over MQTT	15
2.2.4.1. Transporting the Access Token Inside the MQTT CONNECT	15
2.2.4.2. Authentication Using AUTH Property	18
2.2.5. Broker Token Validation	21
2.3. Token Scope and Authorization	22
2.4. Broker Response to Client Connection Request	23
2.4.1. Unauthorized Request and the Optional Authorization Server Discovery	23
2.4.2. Authorization Success	24
3. Authorizing PUBLISH and SUBSCRIBE Packets	24
3.1. PUBLISH Packets from the Publisher Client to the Broker	24
3.2. PUBLISH Packets from the Broker to the Subscriber Clients	25
3.3. Authorizing SUBSCRIBE Packets	25
4. Token Expiration, Update, and Reauthentication	26
5. Handling Disconnections and Retained Messages	27
6. Reduced Protocol Interactions for MQTT v3.1.1	28
6.1. Token Transport	28
6.2. Handling Authorization Errors	30
7. IANA Considerations	31
7.1. TLS Exporter Label Registration	31
7.2. Media Type Registration	31
7.3. ACE OAuth Profile Registration	32
7.4. AIF	33
8. Security Considerations	33
9. Privacy Considerations	34
10. References	35

10.1. Normative References	35
10.2. Informative References	38
Appendix A. Checklist for profile requirements	40
Appendix B. Document Updates	40
Acknowledgments	45
Authors' Addresses	45

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-v3.1.1]. 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 to. 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 Authorization 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 the Clients to the Broker

Clients use the MQTT PUBLISH packet to publish to a topic. The mechanisms specified in this document do not protect the payload of the PUBLISH packet 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 Broker authentication to the MQTT Clients, 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 the 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 [I-D.ietf-httpbis-semantics], using "application/ace+json" content type, and unless otherwise specified, using JSON encoding. The token MAY be an opaque reference to authorization information or JSON Web Token (JWT) [RFC7519]. For JWTs, this document follows [RFC7800] for PoP semantics for JWTs, and the mechanisms for providing and verifying PoP are detailed in Section 2.2. The Client-AS and RS-AS exchanges MAY also use protocols other than HTTP, e.g., Constrained Application Protocol (CoAP) [RFC7252] or MQTT. It is recommended that TLS is used to secure these communication channels between Client-AS and RS-AS. To reduce the protocol memory and bandwidth requirements, implementations MAY also use "application/ace+cbor" content type, and CBOR encoding [RFC8949], and CBOR Web Token (CWT) [RFC8392] and associated PoP semantics. For more information, see Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs) [RFC8747]. A JWT token uses JOSE, while a CWT token uses COSE [RFC8152] for security protection.

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 Authorization 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 to the AS using HTTPS.

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-v3.1.1].

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.

Network Connection

A construct provided by the underlying transport protocol that is being used by MQTT. It connects the Client to the Server. It provides the means to send an ordered, lossless, stream of bytes in both directions. This document uses TLS as transport protocol.

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 Topic Name.

MQTT Control Packet

The MQTT protocol operates by exchanging a series of MQTT Control packets. Each packet is composed of a Fixed Header, a Variable Header (depending on the control packet type), and a Payload.

UTF-8 encoded string

A string prefixed with a two-byte length field that gives the number of bytes in a UTF-8 encoded string itself. Unless stated otherwise, all UTF-8 encoded strings can have any length in the range 0 to 65535 bytes.

Binary Data

Binary Data is represented by a two-byte length field which indicates the number of data bytes, followed by that number of bytes. Thus, the length of Binary Data is limited to the range of 0 to 65535 Bytes.

Variable Byte Integer

Variable Byte Integer is encoded using an encoding scheme that uses a single byte for values up to 127. For larger values, the least significant seven bits of each byte encode the data, and the most significant bit is used to indicate whether there are bytes following in the representation. Thus, each byte encodes 128 values and a "continuation bit". The maximum number of bytes in the Variable Byte Integer field is four.

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 packets (e.g. CONNECT, CONNACK). A Property consists of an Identifier that defines its usage and data type, followed by a value. The Identifier is encoded as a Variable Byte Integer. For example, the "Authentication Data" property uses the 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 packets 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 from the Broker to the Client.

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 packet.

PINGRESP

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

DISCONNECT

The DISCONNECT packet is the final MQTT Control Packet sent from the Client or the Broker. It indicates the reason why the Network Connection is being closed. If the Network Connection is closed without the Client first sending a DISCONNECT packet with Reason Code 0x00 (Normal disconnection) and the Connection has a Will Message, the Will Message is published.

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 packet. Situations in which the Will Message is published include, but are not limited to:

- * An I/O error or network failure detected by the Broker.
- * The Client fails to communicate within the Keep Alive period.
- * The Client closes the Network Connection without first sending a DISCONNECT packet with a Reason Code 0x00 (Normal disconnection).
- * The Broker closes the Network Connection without first receiving a DISCONNECT packet with a Reason Code 0x00 (Normal disconnection).

If the Will Flag is set in the CONNECT flags, then the payload of the CONNECT packet 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 AS and verified by the MQTT Broker. Figure 1 shows the basic protocol flow during connection setup. The token request and response use the token endpoint at the AS, specified for HTTP-based interactions in Section 5.8 of the ACE framework [I-D.ietf-ace-oauth-authz]. Steps (D) and (E) are optional and use the introspection endpoint specified in Section 5.9 of the ACE framework. The discussion in this document assumes that the Client and the Broker use HTTPS to communicate with the 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. Whatever protocol is used for C-AS and Broker-AS communications MUST provide mutual authentication, confidentiality protection, and integrity protection.

If the Client is resource-constrained or does not support HTTPS, a separate Client Authorization Server may carry out the token request on behalf of the Client (Figure 1 (A) and (B)), and later, onboard the Client with the token. The interactions between a Client and its Client Authorization Server for token onboarding and support for MQTT-based token requests at the AS are out of the scope of this document.

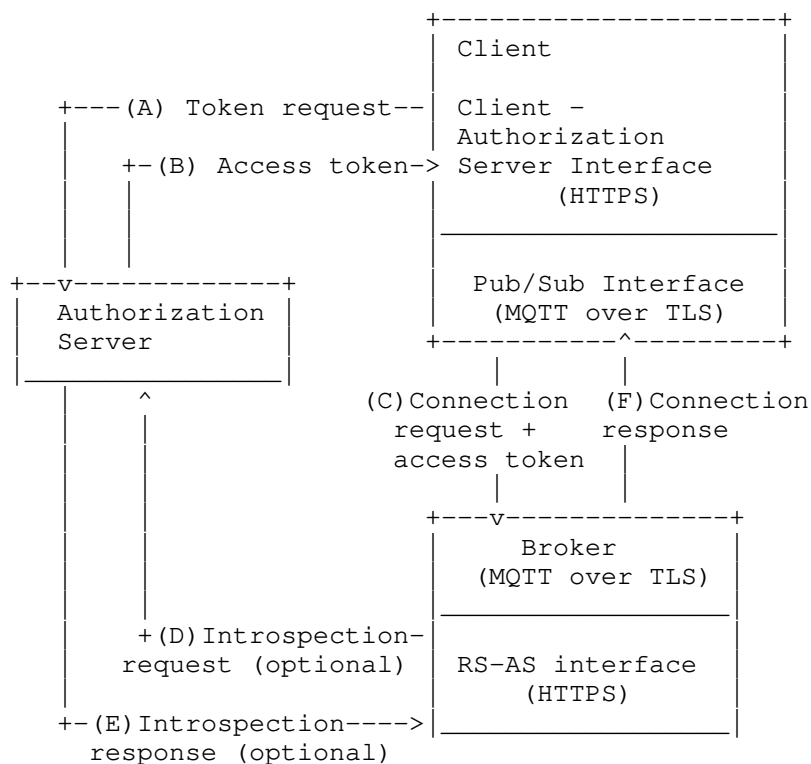


Figure 1: Connection Setup

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 5.8.1 of the ACE framework [I-D.ietf-ace-oauth-authz]. The document follows the procedures defined in Section 3.2.1 of the DTLS profile [I-D.ietf-ace-dtls-authorize] for RPK (Raw Public Keys [RFC7250]), and in Section 3.3.1 of the same document for PSK (Pre-Shared Keys). However, the content type of the request is set to "application/ace+json", and the AS uses JSON in the payload of its responses to the Client and the RS. As explained earlier, implementations MAY also use "application/ace+cbor" content type.

On receipt of the token request, the AS verifies the request. 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 2.3), the AS issues an access token (Figure 1 (B)).

The response includes the parameters described in Section 5.8.2 of the ACE framework [I-D.ietf-ace-oauth-authz]. For RPK, the parameters are as described in Section 3.2.1 of the DTLS profile [I-D.ietf-ace-dtls-authorize]. For PSK, the document follows Section 3.3.1 of the DTLS profile [I-D.ietf-ace-dtls-authorize]. In both cases, if the response contains an "ace_profile" parameter, this 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 (CWTs MAY also be used). The AS includes a "cnf" (confirmation) parameter in the PoP token, to declare that the Client possesses a particular key and RS can cryptographically confirm that the Client has possession of that key, as described in [I-D.ietf-ace-oauth-params].

Note that the contents of the web tokens (including the "cnf" parameter) are to be consumed by the RS and not the Client (the Client obtains the key information in a different manner). The RPK case is handled as described in Section 3.2.1 of the DTLS profile [I-D.ietf-ace-dtls-authorize]. For the PSK case, the referenced procedures apply, with the following exceptions to accommodate JWT and JOSE use. In this case, the AS adds a "cnf" parameter to the access information carrying a JWK (JSON Web Key) [RFC7517] object that contains either the symmetric key itself or a key identifier that can be used by the RS to determine the secret key it shares with the Client. The JWT is created as explained in Section 7 of [RFC7519], and the JWT MUST include JWE [RFC7516]. If CWT/COSE is used this information MUST be inside the "COSE_Key" object, and MUST be encrypted using a "COSE_Encrypt0" structure.

The AS returns error responses for JSON-based interactions following Section 5.2 of [RFC6749]. When CBOR is used, the interactions MUST implement Section 5.8.3 of the ACE framework [I-D.ietf-ace-oauth-authz].

2.2. Client Connection Request to the Broker (C)

2.2.1. Overview of Client-RS Authentication Methods over TLS and MQTT

Unless the Client publishes and subscribes to only public topics, the Client and the Broker MUST perform mutual authentication. The Client MUST authenticate to the Broker either over MQTT or TLS before performing any other action. For MQTT, the options are "None" and "ace". For TLS, the options are "Anon" for an anonymous client, and "Known(RPK/PSK)" for RPK and PSK, respectively. The "None" and "Anon" options do not provide client authentication but can be used either during authentication or in combination with authentication at the other layer. When the Client uses TLS:Anon,MQTT:None, the Client can only publish or subscribe to public topics. Thus, the client authentication procedures involve the following possible combinations:

- * 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 packet 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. As per the ACE framework [I-D.ietf-ace-oauth-authz], a separate step is needed to determine whether the discovered AS URI is authorized to act as an AS.
- * TLS:Known(RPK/PSK),MQTT:none: This specification supports client authentication with TLS with RPK and PSK following the procedures described in DTLS profile [I-D.ietf-ace-dtls-authorize]. For the RPK, the Client MUST have published the token to the "authz-info" topic. For the PSK, the token MAY be published to the "authz-info" topic, or MAY be, alternatively, provided as a "PSK identity" (e.g. an "identity" in the "identities" field in the Client's "pre_shared_key" extension in TLS 1.3).
- * 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 the first choice when working with protected topics. However, MQTT v3.1.1 Clients that do not prefer to overload username and password fields for ACE (as described in Section 6) MAY implement TLS:Known(RPK/PSK),MQTT:none, and consequently TLS:Anon,MQTT:None to submit their 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.

The Client MUST authenticate the Broker 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 an X.509 certificate or an RPK from the Broker against the "rs_cnf" parameter in the token response, which contains information about the public key used by the RS to authenticate if the token type is "pop" and asymmetric keys are used as defined in [I-D.ietf-ace-oauth-params]. 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 the DTLS profile for ACE [I-D.ietf-ace-dtls-authorize]). This profile follows the recommendation of Section 5.10.1 of the ACE framework [I-D.ietf-ace-oauth-authz] and expects that the Broker stores only one token per PoP 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 packet, the Broker returns the error response as a PUBACK, PUBREC, or DISCONNECT packet. If the QoS level is equal to 0, and the token is not valid, or the claims cannot be obtained in the case of an introspected token, the Broker MUST send a DISCONNECT packet with the reason code 0x87 (Not authorized). If the PUBLISH payload does not parse to a token, the Broker MUST send a DISCONNECT with the reason code 0x99 (Payload format invalid).

If the QoS level of the PUBLISH packet is greater than or equal to 1, and the token is not valid, or the claims cannot be obtained in the case of an introspected token, the Broker MUST send the reason code 0x87 (Not authorized) in the PUBACK or PUBREC. If the PUBLISH payload does not parse to a token, the PUBACK/PUBREC reason code is 0x99 (Payload format invalid).

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

2.2.3. Client Authentication over TLS

This document supports TLS with Raw Public Keys (RPK) [RFC7250] and with Pre-Shared Keys (PSK). The TLS session setup follows the DTLS profile for ACE [I-D.ietf-ace-dtls-authorize], as the profile applies to TLS equally well [I-D.ietf-ace-extend-dtls-authorize]. When there are exceptions to the DTLS profile, these are explicitly stated in the document. If TLS 1.2 is used, [RFC7925] describes how TLS can be used for constrained devices, alongside recommended cipher suites. Additionally, TLS 1.2 implementations MUST use the "Extended Main Secret" extension (terminology adopted from [I-D.ietf-tls-rfc8446bis]) to incorporate the handshake transcript into the main secret [RFC7627]. TLS implementations SHOULD use the SNI (Server Name Indication) [RFC6066] and APLN (Application-Layer Protocol Negotiation) [RFC7301] extensions so the TLS handshake authenticates as much of the protocol context as possible.

2.2.3.1. Raw Public Key Mode

This document follows the procedures defined in Section 3.2.2 of the DTLS profile for ACE [I-D.ietf-ace-dtls-authorize] with the following exceptions. The Client MUST upload the access token to the Broker using the method specified in Section 2.2.2 before initiating the handshake.

2.2.3.2. Pre-Shared Key Mode

This document follows the procedures defined in Section 3.3.2 of DTLS profile for ACE [I-D.ietf-ace-dtls-authorize] with the following exceptions.

To use TLS 1.3 with pre-shared keys, the Client utilizes the PSK key extension specified in [RFC8446] using the key conveyed in the "cnf" parameter of the AS response. The same key is bound to the access token in the "cnf" claim. The Client can upload the token as specified in Section 2.2.2 before initiating the handshake. When using a previously uploaded token, the Client MUST indicate during the handshake which previously uploaded access token it intends to use. To do so, it MUST create a "COSE_Key" or "JWK" structure with the "kid" that was conveyed in the "rs_cnf" claim in the token response from the AS and the key type "symmetric". This structure is then included as the only element in the "cnf" structure and the encoded value of that "cnf" structure used as a PSK identity in TLS. As an alternative to the access token upload, the Client can provide the most recent access token, JWT or CWT, as a PSK identity.

In contrast to DTLS profile for ACE [I-D.ietf-ace-dtls-authorize], a Client MAY omit support for the cipher suites TLS_PSK_WITH_AES_128_CCM_8 and TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8. For TLS 1.2, however, a client MUST support TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256 for PSK ([RFC8442]) and TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 for RPK ([RFC8422]), as recommended in [RFC7525] (and adjusted to be a PSK cipher suite as appropriate).

2.2.4. Client Authentication over MQTT

2.2.4.1. Transporting the Access Token Inside the MQTT CONNECT

This section describes how the Client transports the token to the Broker inside the CONNECT packet. If this method is used, the Client TLS connection is expected to be anonymous, and the Broker is authenticated during the TLS connection setup. 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 packets it is allowed to send are DISCONNECT or AUTH that is in response to the Broker AUTH. Similarly, except for a DISCONNECT and AUTH response from the Client, the Broker MUST NOT process any packets before sending a CONNACK.

Figure 2 shows the structure of the MQTT CONNECT packet used in MQTT v5.0. A CONNECT packet 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. Remaining Length is a Variable Byte Integer that represents the number of bytes remaining within the current Control Packet, including data in the Variable Header and the Payload. 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. The Client Identifier identifies the Client to the Broker, and therefore, is unique for each Client. It must be noted that the Client Identifier is an unauthenticated identifier used within the MQTT protocol and so is not bound to the access token.

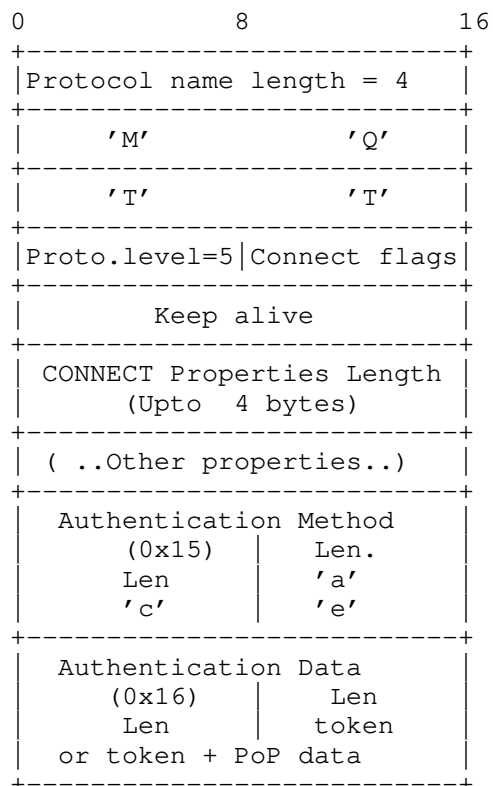


Figure 2: MQTT v5 CONNECT Variable Header with Authentication Method Property for ACE

The CONNECT 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 authorization, i.e., the username and password flags MUST be set to 0. An MQTT v5.0 Broker 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. The Client MAY set the Will Flag as desired (marked as "X" in Figure 3). If the Will Flag is set to 1, 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). The check is performed against the token scope described in Section 2.3. If the Will authorization fails, the connection is refused as described in Section 2.4.1. If the Broker accepts the connection request, the Broker stores the Will message and publishes it when the Network Connection is closed according to Will QoS, and Will retain parameters and MQTT Will management rules. To avoid publishing the 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., that the session does not continue an existing session) by setting the Clean Start Flag to 1 in the CONNECT packet. In this profile, the Client SHOULD always start with a clean session. The Broker MAY also signal that it does not support session continuation by setting Session Expiry Interval to 0 in the CONNACK. 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. In this case, if a CONNECT packet is received with Clean Start set to 0 and there is a Session associated with the Client Identifier, the Broker MUST resume communications

with the Client based on the state from the existing Session. In its response, the Broker MUST set the Session Present flag to 1 in the CONNACK packet to signal session continuation to the Client. The session state stored by the Client and the Broker is described in Section 5.

When reconnecting to a Broker that supports session continuation, the Client MUST still provide a token, in addition to using the same Client Identifier and setting the Clean Start to 0. The Broker MUST still perform PoP 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 MQTT packets that are pending transmission. When a Client connects with a long Session Expiry Interval, the Broker may need to maintain the 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 the Broker MUST associate the Client with a valid token, a Client will only send or receive messages to its 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.

2.2.4.2. 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 Broker 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. Based on the Authentication Data, the Broker MUST support both options below:

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

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

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

Authentication Data Length	Token Length	Token	MAC or Signature (over TLS exporter content)
-------------------------------	--------------	-------	---

Figure 4: Authentication Data for PoP Based on TLS Exporter Content

For this option, the Authentication Data inside the Client's CONNECT 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 Section 7.5 of [RFC8446]) for TLS 1.3. 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 Broker responds with a CONNACK with the appropriate response code. The Client cannot reauthenticate using this method during the same TLS session (see Section 4).

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

Authentication Data Length	Token Length	Token
-------------------------------	--------------	-------

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

Authentication Data Length	RS Nonce (8 bytes)
-------------------------------	-----------------------

Figure 6: Authentication Data for Broker Challenge

For this option, the Broker follows a Broker-generated challenge/response protocol. If the Authentication Data inside the Client's CONNECT contains only the two-byte integer token length and the token (as shown in Figure 5), the Broker MUST respond with an AUTH packet, with the Authenticate Reason Code set to 0x18 (Continue Authentication). The Broker also uses this method if the Authentication Data does not contain a token, but the Broker has a token stored for the connecting Client.

The Broker continues authentication using an AUTH packet that contains the Authentication Method and the Authentication Data. The Authentication Method MUST be set to "ace", and the Authentication Data MUST NOT be empty and MUST contain an 8-byte RS nonce as a challenge for the Client (Figure 6).

+-----+-----+-----+		
Authentication	Client Nonce	MAC or Signature
Data Length	(8 bytes)	(over RS nonce+Client nonce)
+-----+-----+-----+		

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 8-byte 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.

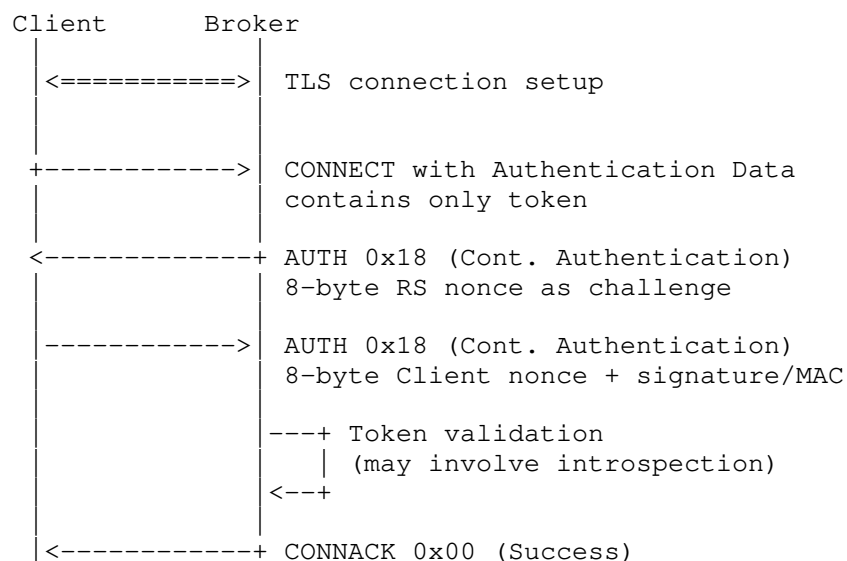


Figure 8: PoP Challenge/Response Flow - Success

2.2.5. Broker Token Validation

The Broker MUST verify the validity of the token either locally (e.g., in the case of a self-contained token) or MAY send a request to the introspection endpoint of the AS (as described for HTTP-based interactions in Section 5.9 of the ACE framework [I-D.ietf-ace-oauth-authz]). The Broker MUST verify the claims in the access token according to the rules set in Section 5.10.1.1 of the ACE framework [I-D.ietf-ace-oauth-authz].

To authenticate the Client, the Broker validates the signature or the MAC, depending on how the PoP protocol is implemented. For self-contained tokens, the Broker MUST process the security protection of the token first, as specified by the respective token format, i.e. a CWT token uses COSE, while a JWT token uses JOSE. For a token-by-reference, the Broker uses the "cnf" structure returned as a result of token introspection as specified in [RFC7519]. HS256 (HMAC-SHA-256) [RFC6234] and Ed25519 [RFC8032] are mandatory to implement for the Broker. The Client MUST implement at least one of them 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.

The Broker MUST check if the access token is still valid, if it is the intended destination (i.e., the audience) of the token, and if the token was issued by an authorized authorization server. If the Client is using TLS RPK mode to authenticate to the Broker, the AS constructs the access token so that the Broker can associate the access token with the Client's public key. The "cnf" claim MUST contain either the Client's RPK or, if the key is already known by the Broker (e.g., from previous communication), a reference to it.

2.3. Token Scope and Authorization

The scope field contains the publish and subscribe permissions for the Client. Therefore, the token or its introspection result MUST be cached to allow a Client's future PUBLISH and SUBSCRIBE messages. During the CONNECT, if the Will Flag is set to 1, the Broker MUST also authorize the publication of the Will Topic and message using the token's scope field. The Broker uses the scope to match against the Topic Name in a PUBLISH packet (including Will Topic in the CONNECT) or a Topic Filter in a SUBSCRIBE packet.

The scope in the token is a single value. For a JWT, the single scope is base64url encoded string with any padding characters removed, which has an internal structure of a JSON array. For a CWT, this information is represented in CBOR. The internal structure follows 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<mqtt-topic-filter, mqtt-permissions>
AIF-Generic<Toid, Tperm> = [* [Toid, Tperm]]
mqtt-topic-filter = tstr ; as per Section 4.7 of MQTT v5.0
mqtt-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]. By default, Wildcard Subscriptions are supported, and so, the topic filter may include special wildcard characters. The multi-level wildcard, "#", matches any number of levels within a topic, and the single-level wildcard, "+", matches one topic level. The Broker MAY signal in the CONNACK explicitly whether wildcard subscriptions are supported by returning a CONNACK property "Wildcard Subscription Available". A value of 0 means that Wildcard Subscriptions are not supported. A value of 1 means Wildcard Subscriptions are supported.

Following this model, 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 scope is empty, the Broker 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. The non-empty scope is used to authorize the Will Topic, if provided, in the CONNECT packet, during connection setup, and if the connection request succeeds, the Topic Names or Topic Filters requested in the future PUBLISH and SUBSCRIBE packets. For the authorization to succeed, the Broker MUST verify that the topic name or filter in question is either an exact match to or a subset of at least one "topic_filter" in the scope.

2.4. Broker 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 packet to the Client.

2.4.1. Unauthorized Request and the Optional Authorization Server Discovery

Authentication can fail for the following reasons:

- * If the Client does not provide a valid token,
- * the Client omits the Authentication Data field and the Broker has no token stored for the Client,
- * the token or Authentication data are malformed, or
- * if the Will flag is set, the authorization checks for the Will topic fails.

The Broker responds with the CONNACK reason code 0x87 (Not Authorized) or any other applicable reason code.

The Broker MAY also trigger AS discovery and include a User Property (identified as property type 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 object containing the mandatory "AS" parameter, and the optional parameters "audience", "kid", "cnonce", and "scope" as defined in Section 5.3 of the ACE framework [I-D.ietf-ace-oauth-authz].

2.4.2. Authorization Success

On success, the reason code of the CONNACK is 0x00 (Success). 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.

Having accepted the connection, the Broker MUST be prepared to store the token during the connection and after disconnection for future use. 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 packets. PUBLISH and SUBSCRIBE packets, which are sent after a connection setup, do not contain access tokens. If the introspection result is not cached, the Broker needs to introspect the saved token for each request. The Broker SHOULD also use a cache timeout to introspect tokens regularly. The timeout value is application-specific and should be chosen to reduce the risk of using stale introspection responses.

3. Authorizing PUBLISH and SUBSCRIBE Packets

Using the cached token or its introspection result, the Broker uses the scope field to match against the Topic Name in a PUBLISH packet, or a Topic Filter in a SUBSCRIBE packet.

3.1. PUBLISH Packets from the Publisher Client to the Broker

On receiving the PUBLISH packet, the Broker MUST use the type of packet (i.e., PUBLISH) and the Topic name in the packet header to match against the scope array items in the cached token or its introspection result. Following the example in Section 2.3, the Client sending a PUBLISH for "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 packet 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 acknowledgment 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).

For all QoS levels, the Broker MAY return 0x80 Unspecified error if they do not want to leak the topic names to unauthorized clients.

3.2. PUBLISH Packets from the Broker to the Subscriber Clients

To forward PUBLISH packets to the subscribing Clients, the Broker identifies all the subscribers that have valid matching topic subscriptions to the Topic name of the PUBLISH packet (i.e., the tokens are valid, and token scopes allow a subscription to this particular Topic name). The Broker forwards the PUBLISH packet to all the valid subscribers.

The Broker MUST NOT forward messages to unauthorized subscribers. To avoid silently dropping messages, the Broker MUST close the network connection and SHOULD inform the affected subscribers. The only way to inform a client, in this case, would be sending a DISCONNECT packet. Therefore, the Broker SHOULD send a DISCONNECT packet with the reason code 0x87 (Not authorized) before closing the network connection to these clients.

3.3. Authorizing SUBSCRIBE Packets

In MQTT, a SUBSCRIBE packet is sent from a Client to the Broker to create one or more subscriptions to one or more topics. The SUBSCRIBE packet may contain multiple Topic Filters. The Topic Filters may include wildcard characters.

On receiving the SUBSCRIBE packet, the Broker MUST use the type of packet (i.e., SUBSCRIBE) and the Topic Filter in the packet header to match against the scope field of the stored token or introspection result. The Topic Filters MUST be an exact match to or be a subset of at least one of the "topic_filter" fields in the scope array found in the Client's token. For example, if the Client sends a subscription request for topic "a/b/*", and has a token that permits "a/*", this is a valid subscription request, as "a/b/*" is a subset of "a/*". (The process is similar to a Broker matching the Topic Name in a PUBLISH packet against the Subscriptions known to the Server.)

As a response to the SUBSCRIBE packet, the Broker issues a SUBACK. 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. The Broker MAY return 0x80 Unspecified error if they do not want to leak the topic names to unauthorized clients. 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 is received or sent. The Broker SHOULD check for token expiration on receiving a PINGREQUEST. 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 5.9 of the ACE framework [I-D.ietf-ace-oauth-authz]. Token expirations may trigger the Broker to send PUBACK, SUBACK and DISCONNECT packets with return code set to "Not authorized". After sending a DISCONNECT, the Network Connection is closed, and no more messages can be sent.

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 packet 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. If re-authenticating during the current TLS session, the Client MUST NOT use the method described in Section 2.2.4.2.1, Proof-of-Possession using a challenge from the TLS session, to avoid re-using the same challenge value from the TLS-Exporter. Note that this means that servers will either need to record in the session ticket or database entry whether the TLS-Exporter-derived challenge was used, or always deny use of the TLS-Exporter-derived challenge for resumed sessions. In TLS 1.3, the resumed connection would have a new exporter value, but the requirement is phrased this way for simplicity. For re-authentications in the same TLS-session, the Client MUST use the challenge-response PoP as defined in Section 2.2.4.2.2. The Broker accepts reauthentication requests if the Client has already submitted a token (may be expired), for which it performed proof-of-possession. 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, if the Session Expiry Interval is set to 0, the Broker doesn't maintain session state but MUST keep the retained messages. If the Broker maintains session state, the state MAY include the 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 the following:

- * Client subscription state,
- * messages with QoS levels 1 and 2, and which have not been completely acknowledged or are 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.

The token/introspection state is not part of the MQTT session state, and PoP validation is required for each new connection, regardless of whether MQTT session continuation is used.

The messages to be retained are indicated to the Broker by setting a RETAIN flag in a PUBLISH packet. This way, the publisher signals 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 packet. The Broker MUST continue publishing the retained messages as long as the associated tokens are valid. In the MQTT standard, if QoS is 0 for the PUBLISH packet, the Broker may discard the retained message any time. For QoS>1, the message expiry interval dictates how long the retained message is kept. However, it is important that the Broker avoids sending messages indefinitely for the Clients that never update their tokens (i.e., the Client connects briefly with a valid token, sends a PUBLISH packet with RETAIN flag set to 1 and QoS>1, disconnects, and never connects again). Therefore, the Broker MUST use the minimum of token expiry and message expiry interval to discard a retained message.

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 packet. 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 (as it has been validated and accepted during CONNECT).

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; The flows described in this section are NOT RECOMMENDED for use by MQTT v5.0 Clients. 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 packet. If the Client provided the token via the "authz-info" topic and will not update the token in the CONNECT packet, it MUST authenticate over TLS. The Broker SHOULD still be prepared to store the Client access token for future use (regardless of the method of transport).

In MQTT v3.1.1, after the Client has published to the "authz-info" topic, the Broker cannot communicate the result of the token validation because PUBACK reason codes or server-side DISCONNECT packets are not supported. In any case, the subsequent TLS handshake would fail without a valid token, which can prompt the Client to obtain a valid token.

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

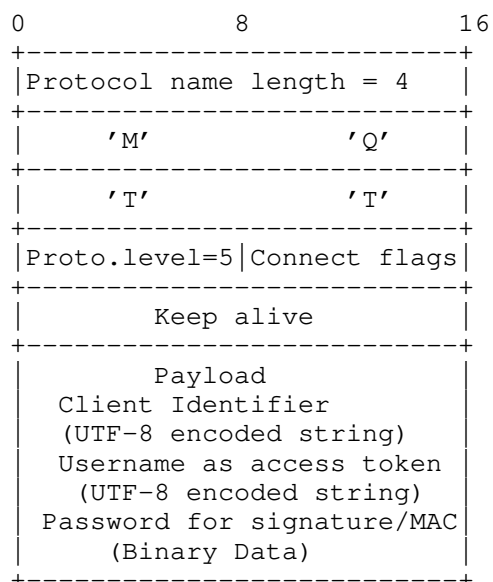


Figure 11: MQTT CONNECT Variable Header Using Username and Password for ACE

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 Client SHOULD set the Clean flag to 1 to always start a new session. If the Clean flag is set to 0, the Broker MUST resume communications with the Client based on the state from the current Session (as identified by the Client Identifier). If there is no Session associated with the Client Identifier, the Broker MUST create a new session. The Broker MUST set the Session Present flag in the CONNACK packet accordingly, i.e., 0 to indicate a clean session to the Client and 1 to indicate session continuation. The Broker MUST still perform PoP 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 action or policies (e.g. defining an automated response based on storage capabilities), and Brokers SHOULD implement administrative 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 packet includes both Username and Password fields. The MQTT Username is a UTF-8 encoded string, and the MQTT Password is Binary Data.

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, i.e., the Username field is a concatenation of 'a', 'c', 'e' and the access token represented as:

```
'U+0061' || 'U+0063' || 'U+0065' || UTF-8(access token)
```

Figure 13: Username in CONNECT

To this end, the access token MUST be base64url encoded, omitting the '=' padding characters [RFC4648].

The password field MUST be set to the keyed message digest (MAC) or signature associated with the access token for PoP. The Client MUST apply the PoP key on the challenge derived from the TLS session as described in Section 2.2.4.2.1.

6.2. Handling Authorization Errors

Error handling is 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 the session state, necessitating that clients make a greater effort to ensure that tokens remain valid and do 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: The tokenless CONNECT attempt MUST fail. This is because the challenge-response based PoP is not possible for MQTT v3.1.1. It is also not possible to support AS discovery since 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.
- * Client-Broker PUBLISH authorization failure: In the case of a failure, it is not possible to return an error in MQTT v3.1.1. Acknowledgment messages only indicate success. In the case of an

authorization error, the Broker MUST ignore the PUBLISH packet and disconnect the Client. Also, as DISCONNECT packets 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.
- * Broker-Client PUBLISH authorization failure: When the Broker is forwarding PUBLISH packets to the subscribed Clients, it may discover that some of the subscribers are no longer authorized due to expired tokens. These token expirations MUST lead to disconnecting the Client rather than silently dropping messages.

7. IANA Considerations

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

7.1. TLS Exporter Label Registration

This document registers "EXPORTER-ACE-MQTT-Sign-Challenge" (introduced in Section 2.2.4.2.1 in this document) in the TLS Exporter Label Registry [RFC8447].

- * Recommended: No
- * DTLS-OK: No
- * Reference: [This document]

7.2. Media Type Registration

This document registers the "application/ace+json" media type for messages of the protocols defined in this document carrying parameters encoded in JSON.

- * Type name: application
- * Subtype name: ace+json
- * Required parameters: N/A
- * Optional parameters: N/A

- * Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type.
- * Security considerations: Section 8 of [this document]
- * Interoperability considerations: none
- * Published specification: [this document]
- * Applications that use this media type: This media type is intended for authorization server-client and authorization server-resource server communication as part of the ACE framework using JSON encoding as specified in [this document].
- * Fragment identifier considerations: none
- * Additional information:
 - Deprecated alias names for this type: none
 - Magic number(s): none
 - File extension(s): none
 - Macintosh file type code(s): none
- * Person & email address to contact for further information: Cigdem Sengul (csengul@acm.org)
- * Intended usage: COMMON
- * Restrictions on usage: none
- * Author: Cigdem Sengul (csengul@acm.org)
- * Change controller: IETF
- * Provisional registration? (standards tree only): no

7.3. ACE OAuth Profile Registration

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

- * Name: mqtt_tls

- * Description: Profile for delegating Client authentication and authorization using MQTT for the Client and Broker (RS) interactions, and HTTP for the AS interactions. TLS is used for confidentiality and integrity protection and server authentication. Client authentication can be provided either via TLS or using in-band PoP validation at the MQTT application layer.
- * CBOR Value: To be assigned by IANA in the (-256, 255) range
- * Reference: [this document]

7.4. AIF

For the media-types application/aif+cbor and application/aif+json defined in Section 5.1 of [I-D.ietf-ace-aif], IANA is requested to register the following entries for the two media-type parameters Toid and Tperm, in the respective sub-registry defined in Section 5.2 of [I-D.ietf-ace-aif] within the "MIME Media Type Sub-Parameter" registry group.

For Toid:

- * Name: mqtt-topic-filter
- * Description/Specification: Topic Filter as defined in Section 2.3.
- * Reference: [[This document]] (Section 2.3)

For Tperm:

- * Name: mqtt-permissions
- * Description/Specification: Permissions for MQTT client as defined in Section 2.3. Tperm is an array of one or more text strings that each have a value of either "pub" or "sub".
- * Reference: [[This document]] (Section 2.3)

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-v3.1.1] 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 Broker 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. The Broker does not cache any tokens that cannot be validated. If a Client's permissions get revoked, but the access token has not expired, the Broker may still grant publish/subscribe to revoked topics. If the Broker caches the token introspection responses, then the Broker SHOULD use a reasonable cache timeout to introspect tokens regularly. The timeout value is application-specific and should be chosen to reduce the risk of using stale introspection responses. When permissions change dynamically, it is expected that AS also follows a reasonable expiration strategy for the access tokens.

The Broker 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 Broker 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 tokens that are not meant to be used, and which the Broker may need to store until the tokens expire.

For MQTT v5.0, when a Client connects with a long Session Expiry Interval, the Broker may need to maintain the 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 Broker 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 Broker is a central trusted party and may forward potentially sensitive information between Clients. The mechanisms defined in this document do not protect the contents of the PUBLISH packet from the Broker, and hence, the content of the PUBLISH packet 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 fields of the packet, such as Topic Name.

10. References

10.1. Normative References

[I-D.ietf-ace-aif]

Bormann, C., "An Authorization Information Format (AIF) for ACE", Work in Progress, Internet-Draft, draft-ietf-ace-aif-07, 15 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-ace-aif-07.txt>>.

[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)", Work in Progress, Internet-Draft, draft-ietf-ace-dtls-authorize-18, 4 June 2021, <<https://www.ietf.org/archive/id/draft-ietf-ace-dtls-authorize-18.txt>>.

[I-D.ietf-ace-extend-dtls-authorize]

Bergmann, O., Mattsson, J. P., and G. Selander, "Extension of the CoAP-DTLS Profile for ACE to TLS", Work in Progress, Internet-Draft, draft-ietf-ace-extend-dtls-authorize-02, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-ace-extend-dtls-authorize-02.txt>>.

[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)", Work in Progress, Internet-Draft, draft-ietf-ace-oauth-authz-46, 8 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-ace-oauth-authz-46.txt>>.

- [I-D.ietf-ace-oauth-params]
Seitz, L., "Additional OAuth Parameters for Authorization in Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-oauth-params-16, 7 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-ace-oauth-params-16.txt>>.
- [I-D.ietf-cose-x509]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates", Work in Progress, Internet-Draft, draft-ietf-cose-x509-08, 14 December 2020, <<https://www.ietf.org/internet-drafts/draft-ietf-cose-x509-08.txt>>.
- [I-D.ietf-httpbis-semantic]
Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantic-19, 12 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-httpbis-semantic-19.txt>>.
- [MQTT-OASIS-Standard-v3.1.1]
Banks, A., Ed. and R. Gupta, Ed., "OASIS Standard MQTT Version 3.1.1 Plus Errata 01", 2015, <<https://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, <<https://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>>.

- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <<https://www.rfc-editor.org/info/rfc7301>>.
- [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <<https://www.rfc-editor.org/info/rfc7516>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <<https://www.rfc-editor.org/info/rfc7517>>.
- [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>>.
- [RFC7627] Bhargavan, K., Ed., Delignat-Lavaud, A., Pironti, A., Langley, A., and M. Ray, "Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension", RFC 7627, DOI 10.17487/RFC7627, September 2015, <<https://www.rfc-editor.org/info/rfc7627>>.
- [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>>.

- [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>>.
- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/info/rfc8152>>.
- [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>>.
- [RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <<https://www.rfc-editor.org/info/rfc8422>>.
- [RFC8442] Mattsson, J. and D. Migault, "ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS 1.2 and DTLS 1.2", RFC 8442, DOI 10.17487/RFC8442, September 2018, <<https://www.rfc-editor.org/info/rfc8442>>.
- [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>>.
- [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, <<https://dx.doi.org/10.1109/SIoT.2014.8>>.

- [I-D.ietf-ace-pubsub-profile]
Palombini, F. and C. Sengul, "Pub-Sub Profile for Authentication and Authorization for Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-pubsub-profile-04, 29 December 2021, <<https://www.ietf.org/archive/id/draft-ietf-ace-pubsub-profile-04.txt>>.
- [I-D.ietf-tls-rfc8446bis]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-rfc8446bis-04, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-tls-rfc8446bis-04.txt>>.
- [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>>.
- [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>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <<https://www.rfc-editor.org/info/rfc7925>>.
- [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>>.
- [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>>.
- [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

Based on the requirements on profiles for the ACE framework [I-D.ietf-ace-oauth-authz], this document fulfills the following:

- * Optional AS discovery: AS discovery is supported with the MQTT v5.0 described in Section 2.2.
- * The communication protocol between the Client and Broker (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.
- * Proof-of-possession protocols: Specified in Section 2.2.4.2; both symmetric and asymmetric keys supported.
- * Content format: For the HTTPS interactions with AS, "application/ace+json".
- * Unique profile identifier: mqtt_tls
- * Token introspection: RS uses HTTPS introspect interface of AS.
- * Token request: Client or its Client AS uses the HTTPS token endpoint of the AS.
- * authz-info endpoint: It MAY be supported using the method described in Section 2.2.2, but is not protected other than by the TLS channel between Client and RS.
- * Token transport: Via "authz-info" topic, or TLS with PSK, provided as a PSK identity, or in MQTT CONNECT packet for both versions of MQTT. The AUTH extensions can also be used for authentication and re-authentication for MQTT v5.0, as described in Section 2.2 and Section 4.

Appendix B. Document Updates

Version 15: Addressing GENART review comments.

Version 11 to 15: Addressing AD review comments.

Version 10 to 11: Clarified the TLS use between RS-AS and Client-AS.

Version 09 to 10: Fixed version issues for references.

Version 08 to 09: Fixed spacing issues and references.

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 Authorization Server.
- * Added artwork to show Authorization 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 Broker will disconnect on almost any error and may not keep session state.
- * Clarified that the Broker does not cache tokens that cannot be validated.

Version 04 to 05:

- * Reorganised Section 2 such that "Unauthorized Request: Authorization 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 Authorization Server Discovery is triggered if a token is not valid 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 authorization 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 authorization 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.

Acknowledgments

The authors would like to thank Ludwig Seitz for his review and his input on the authorization information endpoint; Benjamin Kaduk for his review, insightful comments, and contributions to resolving issues; and Carsten Bormann for his review and revisions to the AIF-MQTT data model. 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
United Kingdom
Email: csengul@acm.org

Anthony Kirby
Oxbotica
1a Milford House, Mayfield Road, Summertown
Oxford
OX2 7EL
United Kingdom
Email: anthony@anthony.org