ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 7 March 2026

M. Tiloca
R. H旦glund
RISE AB
F. Palombini
Ericsson AB

3 September 2025

The Group Object Security for Constrained RESTful Environments (Group OSCORE) Profile of the Authentication and Authorization for Constrained Environments (ACE) Framework
draft-ietf-ace-group-oscore-profile-05

Abstract

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework. The profile uses Group Object Security for Constrained RESTful Environments (Group OSCORE) to provide communication security between a client and one or multiple resource servers that are members of an OSCORE group. The profile securely binds an OAuth 2.0 access token to the public key of the client associated with the private key used by that client in the OSCORE group. The profile uses Group OSCORE to achieve server authentication and proof of possession of the client's private key. Also, it provides proof of the client's membership to the OSCORE group by binding the access token to information from the Group OSCORE Security Context, thus allowing the resource server(s) to verify the client's membership upon receiving a message protected with Group OSCORE from the client. Effectively, the profile enables fine-grained access control paired with secure group communication, in accordance with the Zero Trust principles.

Discussion Venues

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

Discussion of this document takes place on the Authentication and Authorization for Constrained Environments Working Group mailing list (ace@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ace/.

Source for this draft and an issue tracker can be found at https://github.com/ace-wg/ace-group-oscore-profile.

Status of This Memo

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

This Internet-Draft will expire on 7 March 2026.

Copyright Notice

Table of Contents

1. Introduction

A number of applications rely on a group communication model where a client can access a resource hosted by multiple resource servers at once, e.g., over IP multicast. Typical examples include switching of luminaries, actuators control, and distribution of software updates. Secure communication in the group can be achieved by sharing a set of keying material, which is typically provided upon joining the group.

For some of such applications, it may be just fine to enforce access control in a straightforward fashion. That is, any client authorized to join the group, hence to obtain the group keying material, can be also implicitly authorized to perform any action at any resource of any server in the group. An example of application where such

implicit authorization might serve well is a simple lighting scenario, where the lightbulbs are the servers, while the user account on an app on the user's phone is the client. In this case, it might be fine to not require additional authorization evidence from any user account, if it is acceptable that any current group member is also authorized to switch on and off any light, or to check the status of any light.

However, in different instances of such applications, the approach above is not desirable, as different group members are intended to have different access rights to resources of other group members. For instance, enforcing access control in accordance with a more fine-grained approach is required in the two following use cases.

As a first case, an application provides control of smart locks acting as servers in the group, where: a first type of client, e.g., a user account of a child, is allowed to only query the status of the smart locks; while a second type of client, e.g., a user account of a parent, is allowed to both query and change the status of the smart locks. Further similar applications concern the enforcement of different sets of permissions in groups with sensor/actuator devices, e.g., thermostats acting as servers. Also, some group members may even be intended as servers only. Hence, they must be prevented from acting as clients altogether and from accessing resources at other servers in the group, especially when attempting to perform non-safe operations.

As a second case, building automation scenarios often rely on servers that, under different circumstances, enforce different level of priority for processing received commands. For instance, BACnet deployments consider multiple classes of clients, e.g., a normal light switch (C1) and an emergency fire panel (C2). Then, a C1 client is not allowed to override a command from a C2 client, until the latter relinquishes control at its higher priority. That is: i) only C2 clients should be able to adjust the minimum required level of priority on the servers, so rightly locking out C1 clients if needed; and ii) when a server is set to accept only high-priority commands, only C2 clients should be able to perform such commands that are otherwise allowed also to C1 clients. Given the different maximum authority of different clients, fine-grained access control would effectively limit the execution of high- and emergency-priority commands only to devices that are in fact authorized to perform such actions. Besides, it would prevent a misconfigured or compromised device from initiating a high-priority command and lock out normal control.

In the cases above, being a legitimate group member and storing the group keying material is not supposed to imply any particular access rights. Instead, access control to the secure group communication channel and access control to the resource space provided by servers in the group should remain logically separated domains.

This is aligned with the Zero Trust paradigm [NIST-800-207], which focuses on resource protection and builds on the premise that trust is never granted implicitly, but must be continually evaluated. In particular, Zero Trust protections involve "minimizing access to resources (such as data and compute resources and applications/ services) to only those subjects and assets identified as needing access as well as continually authenticating and authorizing the identity and security posture of each access request."

Furthermore, [NIST-800-207] highlights how the Zero Trust goal is to "prevent unauthorized access to data and services coupled with making the access control enforcement as granular as possible", in order to "enforce least privileges needed to perform the action in the request."

As a step in this direction, one can be tempted to introduce a different security group for each different set of access rights. However, this inconveniently results in additional keying material to distribute and manage. In particular, if the access rights pertaining to a node change, this requires to evict the node from the group, after which the node has to join a different group aligned with its new access rights. Moreover, the keying material of both groups would have to be renewed for their current members. Overall, this would have a non negligible impact on operations and performance.

Instead, a fine-grained yet flexible access control model can be enforced within the same group, by using the Authentication and Authorization for Constrained Environments (ACE) framework [RFC9200]. That is, a client has to first obtain authorization credentials in the form of an access token and then upload it to the intended resource server(s) in the group before accessing the target resources hosted therein.

The ACE framework delegates to separate profile documents how to secure communications between the client and the resource servers. However each of the current profiles of ACE defined in [RFC9202][RFC9203][RFC9431][I-D.ietf-ace-edhoc-oscore-profile] relies on a security protocol that cannot be used to protect one-to-many group messages, for example sent over IP multicast.

This document specifies the "coap_group_oscore" profile of the ACE framework, where a client uses the Constrained Application Protocol (CoAP) [RFC7252][I-D.ietf-core-groupcomm-bis] to communicate with one or multiple resource servers that are members of an application group and share a common set of resources. This profile uses Group Object Security for Constrained RESTful Environments (Group OSCORE) [I-D.ietf-core-oscore-groupcomm] as the security protocol to protect messages exchanged between the client and the resource servers. Hence, it requires that both the client and the resource servers have previously joined the same OSCORE group.

That is, this profile describes how access control is enforced for a client after it has joined an OSCORE group, to access resources hosted by other members of that group. The process for joining the OSCORE group through the respective Group Manager as defined in [I-D.ietf-ace-key-groupcomm-oscore] takes place before the process described in this document and is out of the scope of this profile.

The client proves its authorization and access rights to the resource server(s) by using an access token bound to a key (the proof-of-possession key). This profile uses Group OSCORE to achieve server authentication and proof of possession of the client’s private key used in the OSCORE group in question. Note that proof of possession is not achieved through a dedicated protocol element, but instead after the first message exchange protected with Group OSCORE.

Furthermore, this profile provides proof of the client’s membership to the OSCORE group, by binding the access token to information from the pre-established Group OSCORE Security Context, as well as to the client’s authentication credential used in the group and including the client’s public key. This allows the resource server(s) to verify the client’s group membership upon reception of a message protected with Group OSCORE from that client.

OSCORE [RFC8613] specifies how to use CBOR Object Signing and Encryption (COSE) [RFC9052][RFC9053] to secure CoAP messages. Group OSCORE builds on OSCORE to provide secure group communication and ensures source authentication: by means of digital signatures embedded in the protected message (when using the group mode); or by protecting a message with pairwise keying material derived from the asymmetric keys of the two peers exchanging the message (when using the pairwise mode).

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 related to Concise Binary Object Representation (CBOR) [RFC8949], COSE [RFC9052][RFC9053], CoAP [RFC7252], Object Security for Constrained RESTful Environments (OSCORE) [RFC8613], and Group OSCORE [I-D.ietf-core-oscore-groupcomm]. These especially include:

* Group Manager, as the entity responsible for a set of groups where communications among members are secured with Group OSCORE.

* 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 [RFC5280], and C509 certificates [I-D.ietf-cose-cbor-encoded-cert].

  Members of an OSCORE group have an associated authentication credential in the format used within the group. As per Section 2.4 of [I-D.ietf-core-oscore-groupcomm], an authentication credential provides the public key as well as a comprehensive set of information related to the public key algorithm, including, e.g., the elliptic curve used (when applicable).

Readers are also expected to be familiar with the terms and concepts described in the ACE framework for authentication and authorization [RFC9200], as well as in the OSCORE profile of ACE [RFC9203]. 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).

Note that the term "endpoint" is used here following its OAuth definition [RFC6749], aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. The CoAP definition, which is "[a]n entity participating in the CoAP protocol" [RFC7252], is not used in this document.

Additionally, this document makes use of the following terminology.

* Pairwise-only group: an OSCORE group that uses only the pairwise mode of Group OSCORE (see Section 8 of [I-D.ietf-core-oscore-groupcomm]).

Examples throughout this document are expressed in CBOR diagnostic notation as defined in Section 8 of [RFC8949] and Appendix G of [RFC8610]. Diagnostic notation comments are often used to provide a textual representation of the parameters' keys and values.

In the CBOR diagnostic notation used in this document, constructs of the form e'SOME_NAME' are replaced by the value assigned to SOME_NAME in the CDDL model shown in Figure 7 of Appendix B. For example, {e'context_id_param': h'abcd0000', e'salt_input_param': h'00'} stands for {71: h'abcd0000', 72: h'00'}.

Note to RFC Editor: Please delete the paragraph immediately preceding this note. Also, in the CBOR diagnostic notation used in this document, please replace the constructs of the form e'SOME_NAME' with the value assigned to SOME_NAME in the CDDL model shown in Figure 7 of Appendix B. Finally, please delete this note.

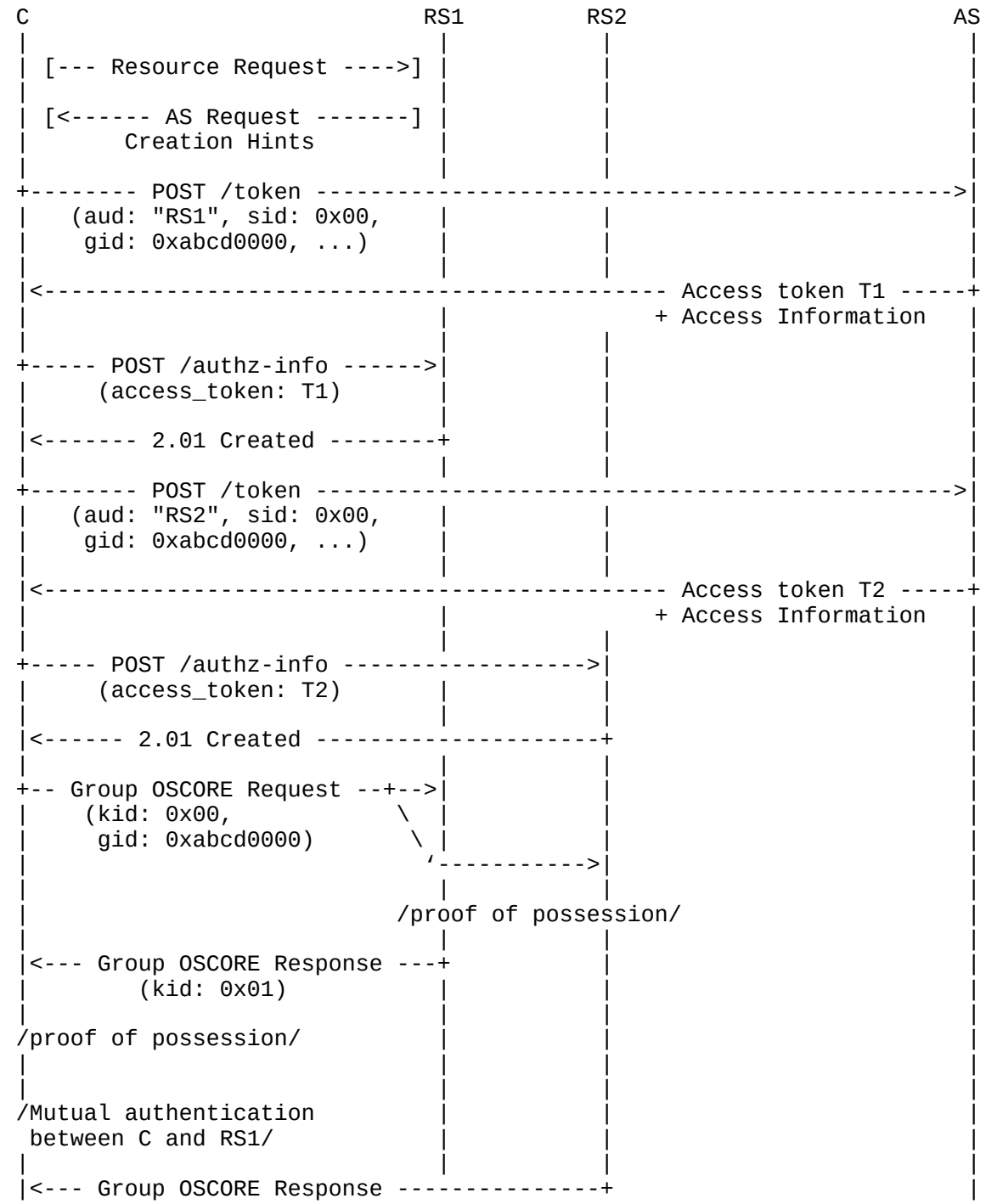
2. Protocol Overview

This section provides an overview of this profile, i.e., of how to use the ACE framework for authentication and authorization [RFC9200] when communications between a client and one or more resource servers are secured using Group OSCORE [I-D.ietf-core-oscore-groupcomm].

Note that this profile of ACE describes how access control can be enforced for a node after it has joined an OSCORE group, to access resources hosted by other members in that group.

In particular, the process of joining the OSCORE group through the respective Group Manager as defined in [I-D.ietf-ace-key-groupcomm-oscore] must take place before the process described in this document and is out of the scope of this profile.

An overview of the protocol flow for this profile is shown in Figure 1, where it is assumed that both the resource servers RS1 and RS2 are associated with the same authorization server AS. It is also assumed that the client C as well as RS1 and RS2 have previously joined an OSCORE group with Group Identifier (gid) 0xabcd0000 and that they got assigned Sender ID (sid) 0x00, 0x01, and 0x02 in the group, respectively. The names of messages coincide with those of [RFC9200] when applicable. Messages in square brackets are optional.


C
RS1
RS2
AS
[--- Resource Request ---->]
[<------ AS Request -------]
Creation Hints
POST /token
(aud: "RS1", sid: 0x00,
gid: 0xabcd0000, ...)
Access token T1
+ Access Information
POST /authz-info
(access_token: T1)
2.01 Created
POST /token
(aud: "RS2", sid: 0x00,
gid: 0xabcd0000, ...)
Access token T2
+ Access Information
POST /authz-info
(access_token: T2)
2.01 Created
Group OSCORE Request
(kid: 0x00,
gid: 0xabcd0000)
/proof of possession/
Group OSCORE Response
(kid: 0x01)
/proof of possession/
/Mutual authentication
between C and RS1/
Group OSCORE Response

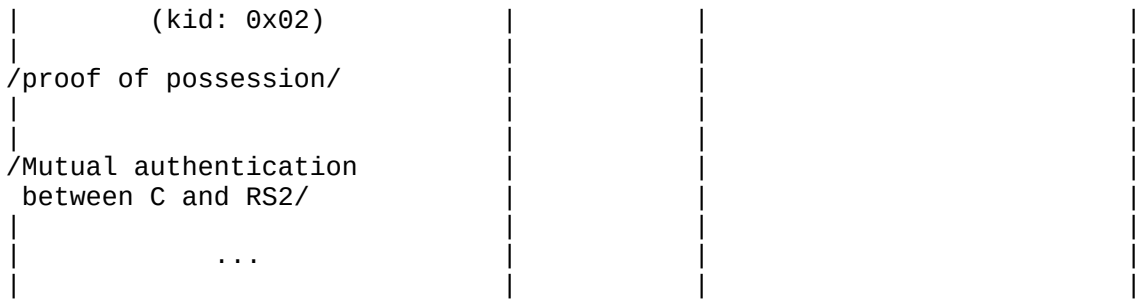


Figure 1: Protocol Overview

2.1. Pre-Conditions

Using Group OSCORE and this profile requires that both the client and the resource servers have previously joined the same OSCORE group. This especially includes the derivation of the Group OSCORE Security Context and the assignment of unique Sender IDs to use in the group. Nodes can join the OSCORE group through the respective Group Manager by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore], which is also based on ACE.

After the client and resource servers have joined the group, this profile provides access control for accessing resources on those resource servers, by securely communicating with Group OSCORE.

As a pre-requisite for this profile, the client has to have successfully joined the OSCORE group where also the resource servers (RSs) are members. Depending on the limited information initially available, the client may have to first discover the exact OSCORE group used by the RSs for the resources of interest, e.g., by using the approach defined in [I-D.tiloca-core-oscore-discovery].

2.2. Requesting an Access Token

This profile requires that the client requests an access token from the AS for the resource(s) that it wants to access at the RS(s), by using the token endpoint as specified in Section 5.8 of [RFC9200].

In general, different RSs can be associated with different authorization servers, even if the RSs are members of the same OSCORE group. However, assuming proper configurations and trust relationships, it is possible for multiple RSs associated with the same AS to be part of a single audience (i.e., a group-audience, see Section 6.9 of [RFC9200]). In such a case, the client can request a single access token intended for the group-audience, hence to all the RSs included therein. A particular group-audience might be defined as including all the RSs in the OSCORE group.

In the access token request to the AS, the client MUST include the Group Identifier of the OSCORE group, as well as its own Sender ID and authentication credential used in that group. The AS MUST include these pieces of information in the access token.

In the access token request, the client can also include a proof-of-possession (PoP) evidence to prove possession of the private key corresponding to its own authentication credential to the AS. The PoP evidence is computed over a PoP input uniquely related to the secure communication association between the client and the AS. Including the PoP evidence is OPTIONAL under particular circumstances and is REQUIRED otherwise (see Section 3.2).

If the request from the client is granted, then the AS can send back the issued access token in the access token response to the client, or instead upload the access token directly to the RS as per the Short Distribution Chain (SDC) workflow defined in [I-D.ietf-ace-workflow-and-params]. This document focuses on the former option (also shown in the example in Figure 1), while the latter option is not detailed further here.

The access token request and response exchanged between the client and the AS MUST be confidentiality-protected and MUST ensure authenticity. In this profile, it is RECOMMENDED to use OSCORE [RFC8613] between the client and the AS, to reduce the number of libraries that the client has to support. Other protocols fulfilling the security requirements defined in Sections 5 and 6 of [RFC9200] MAY alternatively be used, such as TLS [RFC8446] or DTLS [RFC9147].

2.3. Access Token Uploading

After having retrieved the access token from the AS, the client uploads the access token to the RS, by sending a POST request to the authz-info endpoint and using the mechanisms specified in Section 5.10 of [RFC9200]. When using this profile, the communication that C has with the authz-info endpoint is not protected.

If the access token is valid, the RS replies to the POST request with a 2.01 (Created) response. Also, the RS associates the received access token with the Group OSCORE Security Context identified by the Group Identifier that is specified in the access token (see Section 2.1.3 of [I-D.ietf-core-oscore-groupcomm]). In practice, the RS maintains a collection of Security Contexts with associated authorization information, for all the clients that it is currently communicating with. The authorization information is a policy that is used as input when processing requests from those clients.

Finally, the RS stores the association between i) the authorization information from the access token; and ii) the Group Identifier of the OSCORE group together with the Sender ID and the authentication credential of the client in that group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]). This binds the access token to the Group OSCORE Security Context of the OSCORE group.

Finally, when the client communicates with the RS using the Group OSCORE Security Context, the RS verifies that the client is a legitimate member of the OSCORE group and especially the exact group member with the same Sender ID associated with the access token. This occurs when verifying a request protected with Group OSCORE, since the request includes the client’s Sender ID and either it embeds a signature computed also over that Sender ID (if protected with the group mode), or it is protected by means of pairwise symmetric keying material derived from the asymmetric keys of the two peers (if protected with the pairwise mode).

The above has considered an access token intended for a single RS. However, as discussed in Section 2.2, an access token can be intended for a group-audience including multiple RSs in the OSCORE group. In such a case, the client could efficiently upload the access token to many or all of those RSs at once (e.g., over IP multicast), after which each RS individually performs the same steps described above.

2.4. Secure Communication

The client can send a request protected with Group OSCORE [I-D.ietf-core-oscore-groupcomm] to the RS. This can be a unicast request targeting the RS, or a one-to-many group request (e.g., over IP multicast) targeting the OSCORE group where the RS is also a member.

To this end, the client uses the Group OSCORE Security Context already established upon joining the OSCORE group (e.g., by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore]), unless it maintains a more recent Security Context that has been established in the group as a result of a group rekeying (see Section 12.2 of [I-D.ietf-core-oscore-groupcomm]).

The RS may send a response back to the client, also protecting it with Group OSCORE.

3. Client-AS Communication

This section details the access token request that the client sends to the token endpoint of the AS, as well as the related access token response.

The access token MUST be bound to the public key of the client as proof-of-possession (PoP) key, which is included in the client's authentication credential specified in the 'cnf' claim of the access token.

3.1. Preliminary Operations

The following considers a client that is a member of an OSCORE group G with GID* as current Group Identifier (GID), within which the client currently has SID* as Sender ID and uses a public authentication credential AUTH_CRED_C that specifies the PoP key K.

The client MUST perform the following steps, before requesting an access token to be bound to AUTH_CRED_C (hence to the PoP key K) and to be associated with the client's membership in group G through the values GID* and SID*.

1. The client checks whether it is a member of any two OSCORE groups G1 and G2 such that all the following conditions hold.

   * Both groups have GID* as current GID.

   * The client uses SID* as Sender ID in both groups.

   * The client uses the same AUTH_CRED_C in both groups.

2. If such two groups G1 and G2 are found at Step 2, then the client moves to Step 3. Otherwise, the client terminates this algorithm and proceeds with requesting the access token as defined in Section 3.2.

3. The client can choose to terminate this algorithm and perform it again later on.

   Alternatively, the client can alter its current group memberships, in order to ensure that two groups like G1 and G2 cannot be determined. To this end, the client has two available options.

   * The client leaves some of the OSCORE groups that could be determined as groups like G1 and G2 (see Section 9.11 of [I-D.ietf-ace-key-groupcomm-oscore]).

   * The client obtains a new Sender ID in some of the OSCORE groups that could be determined as groups like G1 and G2. To this end, the client can request a new Sender ID in a group to the Group Manager responsible for that group (see Section 9.2 of [I-D.ietf-ace-key-groupcomm-oscore]), or re-join a group thereby obtaining a new Sender ID in that group (see Section 6 of [I-D.ietf-ace-key-groupcomm-oscore]).

   Finally, the client moves to Step 1.

3.2. C-to-AS: POST to Token Endpoint

The Client-to-AS request is specified in Section 5.8.1 of [RFC9200]. The client MUST send this POST request to the token endpoint over a secure channel that guarantees authentication, message integrity, and confidentiality.

The POST request is formatted as the analogous Client-to-AS request in the OSCORE profile of ACE (see Section 3.1 of [RFC9203]), with the following additional parameters that MUST be included in the payload.

* 'context_id', defined in Section 3.2.1 of this document. This parameter specifies the GID, i.e., the ID Context of an OSCORE group that includes as members both the client and the RS(s) in the audience for which the access token is asked to be issued. In particular, the client wishes to communicate with the RS(s) in that audience using the Group OSCORE Security Context associated with that OSCORE group.

* 'salt_input', defined in Section 3.2.2 of this document. This parameter includes the Sender ID that the client has in the OSCORE group whose GID is specified in the 'context_id' parameter above.

* 'req_cnf', defined in Section 3.1 of [RFC9201]. This parameter follows the syntax from Section 3.1 of [RFC8747] and its inner confirmation value specifies the authentication credential

AUTH_CRED_C that the client uses in the OSCORE group. The public key included in the authentication credential will be used as the PoP key bound to the access token.

At the time of writing this specification, acceptable formats of authentication credentials in Group OSCORE are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC5280], 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 compiled in Section 2.4 of [I-D.ietf-core-oscore-groupcomm]. In particular, an authentication credential has to explicitly include the public key as well as a comprehensive set of information related to the public key algorithm, including, e.g., the elliptic curve used (when applicable).

Note that C might have previously uploaded AUTH_CRED_C to the Group Manager as provided within a chain or a bag (e.g., as the end-entity certificate in a chain of certificates), by specifying it in the 'client_cred' parameter of a Join Request or of an Authentication Credential Update Request sent to the Group Manager (see Sections 6.1 and 9.4 of [I-D.ietf-ace-key-groupcomm-oscore]). In such a case, the inner confirmation value of the 'req_cnf' parameter MUST specify AUTH_CRED_C as provided within the same chain or bag.

[As to CWTs and CCSs, the CWT Confirmation Methods 'kcwt' and 'kccs' are under pending registration requested by draft-ietf-ace-edhoc-oscore-profile.]

[As to X.509 certificates, the CWT Confirmation Methods 'x5bag' and '5chain' are under pending registration requested by draft-ietf-ace-edhoc-oscore-profile.]

[As to C509 certificates, the CWT Confirmation Methods 'c5b'and 'c5c' are under pending registration requested by draft-ietf-ace-edhoc-oscore-profile.]

Furthermore, the payload of the request can include exactly one of the two following parameters, specifying a proof-of-possession (PoP) evidence computed by the client.

* 'client_cred_verify', defined in Section 3.2.3 of this document, specifying the client's PoP evidence as a signature, which is computed as defined later in this section. This parameter MUST NOT be included if the OSCORE group is a pairwise-only group.

* 'client_cred_verify_mac', defined in Section 3.2.4 of this document, specifying the client's PoP evidence as a MAC, which is computed as defined later in this section. This parameter MUST NOT be included if the OSCORE group is not a pairwise-only group.

The PoP evidence can be used by the AS to achieve proof of possession of the client's private key, i.e., to verify that the client indeed owns the private key associated with the public key within AUTH_CRED_C.

When preparing the POST request, the client might know that the AS has previously achieved proof of possession of the private key in question. In such a case, it is OPTIONAL for the client to compute the PoP evidence and to specify it in the 'client_cred_verify' or 'client_cred_verify_mac' parameter of the POST request.

If the client believes that the AS has not previously achieved proof of possession of the private key in question or that such proof was achieved but does not hold anymore, then the client MUST compute the PoP evidence as defined below and MUST specify it in the 'client_cred_verify' or 'client_cred_verify_mac' parameter of the POST request.

In order to compute the PoP evidence, the client MUST use as PoP input the byte representation of an information that uniquely represents the secure communication association between the client and the AS. It is RECOMMENDED that the client uses the following as PoP input.

* If the client and the AS communicate over TLS 1.2 [RFC5246] or DTLS 1.2 [RFC6347], the PoP input is an exporter value computed as defined in Section 4 of [RFC5705], using the following inputs:

  - The exporter label "EXPORTER-ACE-PoP-Input-Client-AS" registered in Section 12.5 of this document.

  - The empty 'context value', i.e., a 'context value' of zero-length.

  - 32 as length value in bytes.

* If the client and the AS communicate over TLS 1.3 [RFC8446] or DTLS 1.3 [RFC9147], the PoP input is an exporter value computed as defined in Section 7.5 of [RFC8446], using the following inputs:

  - The exporter label "EXPORTER-ACE-PoP-Input-Client-AS" registered in Section 12.5 of this document.

- The empty 'context_value', i.e., a 'context_value' of zero-length.

- 32 as 'key_length' in bytes.

* If the client and the AS communicate over OSCORE [RFC8613], the PoP input is the output PRK of an HKDF-Extract step [RFC5869], i.e., PRK = HMAC-Hash(salt, IKM).

  In particular, given the OSCORE Security Context CTX shared between the client and the AS, then the following applies.

  - 'salt' takes (x1 | x2), where | denotes byte string concatenation, while x1 and x2 are defined as follows.

  - x1 is the binary representation of a CBOR data item. If CTX does not specify an OSCORE ID Context, the CBOR data item is the CBOR simple value null (0xf6). Otherwise, the CBOR data item is a CBOR byte string, with value the OSCORE ID Context specified in CTX.

  - x2 is the binary representation of a CBOR byte string. The value of the CBOR byte string is the OSCORE Sender ID of the client, which the client stores in its Sender Context of CTX and the AS stores in its Recipient Context of CTX.

  - 'IKM' is the OSCORE Master Secret specified in CTX.

  - The used HKDF is the HKDF Algorithm specified in CTX.

  The following shows an example of input to the HMAC-Hash() function.

  On the client side, the OSCORE Security Context shared with the AS includes:

  ID Context: 0x37cbf3210017a2d3 (8 bytes)

  Sender ID: 0x01 (1 byte)

  Master Secret: 0x0102030405060708090a0b0c0d0e0f10 (16 bytes)

  Then, the following holds.

x1 (Raw value) (8 bytes)
0x37cbf3210017a2d3

x1 (CBOR Data Item) (9 bytes)
0x4837cbf3210017a2d3

x2 (Raw value) (1 bytes)
0x01

x2 (CBOR Data Item) (2 bytes)
0x4101

salt (11 bytes)
0x4837cbf3210017a2d34101

IKM (16 bytes)
0x0102030405060708090a0b0c0d0e0f10

After that, the client computes the PoP evidence as follows.

* If the OSCORE group is not a pairwise-only group, the PoP evidence MUST be a signature. The client computes the signature by using the same private key and signature algorithm that it uses for signing messages in the OSCORE group. The client's private key is the one associated with the client's authentication credential used in the OSCORE group and specified in the 'req_cnf' parameter above.

* If the OSCORE 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.

  - salt takes as value the empty byte string.

  - IKM is computed as a cofactor Diffie-Hellman shared secret (see Section 5.7.1.2 of [NIST-800-56A]), using the ECDH algorithm that is used as Pairwise Key Agreement Algorithm in the OSCORE group. The client uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the AS. For X25519 and X448, the procedure is described in Section 5 of [RFC7748].

The client's private key is the one associated with the client's authentication credential used in the OSCORE group and specified in the 'req_cnf' parameter above. The client may obtain the Diffie-Hellman public key of the AS during its registration process at the AS.

- info takes as value the PoP input.

- L is equal to 8, i.e., the size of the MAC, in bytes.

An example of the POST request is shown in Figure 2.

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: 19 (application/ace+cbor)
Payload:
{
  / audience /        5 : "tempSensor4711",
  / scope /           9 : "read",
    e'context_id_param' : h'abcd0000',
    e'salt_input_param' : h'00',
  e'client_cred_verify' : h'c5a6...f100' / elided for brevity /,
  / req_cnf /         4 : {
    e'kccs' : {
      / sub / 2 : "42-50-31-FF-EF-37-32-39",
      / cnf / 8 : {
        / COSE_Key / 1 : {
          / kty /  1 : 2 / EC2 /,
          / crv / -1 : 1 / P-256 /,
          / x /   -2 : h'd7cc072de2205bdc1537a543d53c60a6
                          acb62eccd890c7fa27c9e354089bbe13',
          / y /   -3 : h'f95e1d4b851a2cc80fff87d8e23f22af
                          b725d535e515d020731e79a3b4e47120'
        }
      }
    }
  }
}
```

Figure 2: Example C-to-AS POST /token Request for an Access Token Bound to an Asymmetric Key

In the example above, the client specifies that its authentication credential in the OSCORE group is the CCS shown in Figure 3.

```
{
  / sub / 2 : "42-50-31-FF-EF-37-32-39",
  / cnf / 8 : {
    / COSE_Key / 1 : {
      / kty /  1 : 2 / EC2 /,
      / crv / -1 : 1 / P-256 /,
      / x /   -2 : h'd7cc072de2205bdc1537a543d53c60a6
                    acb62eccd890c7fa27c9e354089bbe13',
      / y /   -3 : h'f95e1d4b851a2cc80fff87d8e23f22af
                    b725d535e515d020731e79a3b4e47120'
    }
  }
}
```

Figure 3: Example of client Authentication Credential as CWT Claims Set (CCS)

[

TODO: Specify how C requests a new access token that dynamically updates its access rights. (See Section 3.3.1 for pre-requirements and a high-level direction)

]

3.2.1. 'context_id' Parameter

The 'context_id' parameter is an OPTIONAL parameter of the access token request message defined in Section 5.8.1 of [RFC9200]. This parameter provides a value that the client wishes to use with the RS as a hint for a security context. Its exact content is profile specific.

3.2.2. 'salt_input' Parameter

The 'salt_input' parameter is an OPTIONAL parameter of the access token request message defined in Section 5.8.1 of [RFC9200]. This parameter provides a value that the client wishes to use as part of a salt with the RS, for deriving cryptographic keying material. Its exact content is profile specific.

3.2.3. 'client_cred_verify' Parameter

The 'client_cred_verify' parameter is an OPTIONAL parameter of the access token request message defined in Section 5.8.1. of [RFC9200]. This parameter provides a signature computed by the client to prove the possession of its own private key.

3.2.4. 'client_cred_verify_mac' Parameter

The 'client_cred_verify_mac' parameter is an OPTIONAL parameter of the access token request message defined in Section 5.8.1. of [RFC9200]. This parameter provides a Message Authentication Code (MAC) computed by the client to prove the possession of its own private key.

3.3. AS-to-C: Response

After having verified the POST request to the token endpoint and that the client is authorized to obtain an access token corresponding to its access token request, the AS proceeds as defined below.

In the following, an authentication credential is denoted as "confirmed" if and only if the AS has achieved proof of possession of the private key associated with the public key of that authentication credential and such proof still holds. Otherwise, an authentication credential is denoted as "non confirmed".

If the access token request specifies neither the 'client_cred_verify' parameter nor the 'client_cred_verify_mac' parameter, then the AS performs the following steps.

* The AS considers the authentication credential AUTH_CRED_C specified in the 'req_cnf' parameter of the access token request.

* If the AS currently knows AUTH_CRED_C as "confirmed", then the AS considers proof of possession of the client's private key to be achieved and it takes no further actions in this respect.

  The AS might already have achieved proof of possession when establishing a secure communication association with the client, or when processing a previous access token request conveying the same AUTH_CRED_C.

  Alternatively, a further entity in a trust relationship with the AS might have already achieved proof of possession of the private key and informed the AS about that. Building on that trust relationship, the AS considered AUTH_CRED_C to be "confirmed" from then on.

* If the AS does not currently know AUTH_CRED_C as "confirmed", then the AS MUST consider the access token request to be invalid.

If both the 'client_cred_verify' and 'client_cred_verify_mac' parameters are present, then the AS MUST consider the access token request to be invalid.

If the access token request specifies either the 'client_cred_verify' parameter or the 'client_cred_verify_mac' parameter, then the AS MUST verify the proof-of-possession (PoP) evidence specified therein. In particular, the AS proceeds as follows.

* As PoP input, the AS uses the same value that the client used (see Section 3.2).

* As public key of the client, the AS uses the one included in the authentication credential AUTH_CRED_C that is specified in the 'req_cnf' parameter of the access token request.

  This requires the AS to support the format of AUTH_CRED_C, i.e., the format of authentication credential that is used in the OSCORE group where the client uses that authentication credential.

  Practically, this is not an issue, since the same format is used by RSs in that group and an RS supporting this profile is expected to be registered only at an AS that supports the formats of authentication credential that the RS supports.

* If the access token request includes the 'client_cred_verify' parameter, this specifies the PoP evidence as a signature. Then, the AS verifies the signature by using the public key of the client.

  This requires the AS to support the signature algorithm and curve (when applicable) that are used in the OSCORE group where the client uses the authentication credential AUTH_CRED_C that is specified in the 'req_cnf' parameter of the access token request.

  Practically, this is not an issue, since the same algorithm and curve (when applicable) are used by RSs in that group and an RS supporting this profile is expected to be registered only at an AS that supports the signature algorithms and curves (when applicable) that the RS supports.

* If the access token request includes the 'client_cred_verify_mac' parameter, this specifies the PoP evidence as a Message Authentication Code (MAC).

Then, the AS recomputes the MAC through the same process taken by the client when preparing the value of the 'client_cred_verify_mac' parameter for the access token request (see Section 3.2), with the difference that the AS uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the client. The verification succeeds if and only if the recomputed MAC is equal to the MAC conveyed as PoP evidence in the access token request.

This requires the AS to support the ECDH algorithm that is used as Pairwise Key Agreement Algorithm in the OSCORE group where the client uses the authentication credential AUTH_CRED_C that is specified in the 'req_cnf' parameter of the access token request.

Practically, this is not an issue, since the same ECDH algorithm is used by RSs in that group and an RS supporting this profile is expected to be registered only at an AS that supports the ECDH algorithms that the RS supports.

If the verification of the PoP evidence succeeds, then the AS considers AUTH_CRED_C to be "confirmed" from then on.

Instead, if the verification of the PoP evidence fails, then the AS MUST consider the access token request to be invalid. Also, the AS MUST consider AUTH_CRED_C to be "non confirmed" from then on, until the AS achieves again proof of possession of the client's private key.

If the access token request was invalid or not authorized, then the AS MUST return an error response as described in Section 5.8.3 of [RFC9200].

Instead, if all verifications are successful, the AS responds as defined in Section 5.8.2 of [RFC9200]. In particular:

* The AS can signal that the use of Group OSCORE is REQUIRED for a specific access token by including the 'ace_profile' parameter with the value "coap_group_oscore" in the access token response. The client MUST use Group OSCORE towards all the resource servers for which this access token is valid. Usually, it is assumed that constrained devices will be pre-configured with the necessary profile, so that this kind of profile signaling can be omitted.

* The AS MUST NOT include the 'rs_cnf' parameter defined in [RFC9201]. In general, the AS may not be aware of the authentication credentials (and public keys included thereof) that the RSs use in the OSCORE group. Also, the client is able to retrieve the authentication credentials of other group members

from the responsible Group Manager, both upon joining the group or later on as a group member, as defined in [I-D.ietf-ace-key-groupcomm-oscore].

* According to this document, the AS includes the 'access_token' parameter specifying the issued access token in the access token response. The alternative Short Distribution Chain (SDC) workflow where the access token is uploaded by the AS directly to the RS is described in [I-D.ietf-ace-workflow-and-params].

The AS MUST include the following information as metadata of the issued access token. The use of CBOR web tokens (CWT) as specified in [RFC8392] is RECOMMENDED.

* The profile "coap_group_oscore". If the access token is a CWT, this is specified in the 'ace_profile' claim of the access token, as per Section 5.10 of [RFC9200].

* The salt input specified in the 'salt_input' parameter of the access token request. If the access token is a CWT, the content of the 'salt_input' parameter MUST be specified in the 'salt_input' claim of the access token defined in Section 3.3.3 of this document.

* The Context ID input specified in the 'context_id' parameter of the access token request. If the access token is a CWT, the content of the 'context_id' parameter MUST be specified in the 'context_id' claim of the access token, defined in Section 3.3.2 of this document.

* The authentication credential that the client uses in the OSCORE group, as specified in the 'req_cnf' parameter of the access token request (see Section 3.2).

  If the access token is a CWT, the client's authentication credential MUST be specified in the 'cnf' claim, which follows the syntax from Section 3.1 of [RFC8747].

Figure 4 shows an example of such an AS response. The access token has been truncated for readability.

```
Header: Created (Code=2.01)
Content-Format: 19 (application/ace+cbor)
Payload:
{
  / access_token / 1 : h'8343a1010aa2044c...00', / elided for brevity /
  / ace_profile / 38 : e'coap_group_oscore',
  / expires_in /   2 : 3600
}
```

Figure 4: Example AS-to-C Access Token Response with the Group OSCORE Profile

Figure 5 shows an example CWT Claims Set, containing the client's public key in the group (as PoP key), as specified by the inner confirmation value in the 'cnf' claim.

```
{
  / aud /           3 : "tempSensorInLivingRoom",
  / iat /           6 : 1719820800,
  / exp /           4 : 2035353600,
  / scope /         9 : "temperature_g firmware_p",
  e'context_id_claim' : h'abcd0000',
  e'salt_input_claim' : h'00',
  / cnf /           8 : {
    e'kccs' : {
      / sub / 2 : "42-50-31-FF-EF-37-32-39",
      / cnf / 8 : {
        / COSE_Key / 1 : {
          / kty /  1 : 2 / EC2 /,
          / crv / -1 : 1 / P-256 /,
          / x /   -2 : h'd7cc072de2205bdc1537a543d53c60a6
                         acb62eccd890c7fa27c9e354089bbe13',
          / y /   -3 : h'f95e1d4b851a2cc80fff87d8e23f22af
                         b725d535e515d020731e79a3b4e47120'
        }
      }
    }
  }
}
```

Figure 5: Example CWT Claims Set

The same CWT Claims Set as in Figure 5 and encoded in CBOR is shown in Figure 6, using the value abbreviations defined in [RFC9200] and [RFC8747]. The bytes in hexadecimal are reported in the first column, while their corresponding CBOR meaning is reported after the "#" sign on the second column, for easiness of readability.

Editor's note: it should be checked (and in case fixed) that the values used below (which are not yet registered) are the final values registered by IANA.

```
A7                                      # map(7)
   03                                   # unsigned(3)
   76                                   # text(22)
      74656D7053656E736F72496E4C6976696E67526F6F6D
      # "tempSensorInLivingRoom"
   06                                   # unsigned(6)
   1A 66826200                          # unsigned(1719820800)
   04                                   # unsigned(4)
   1A 79510800                          # unsigned(2035353600)
   09                                   # unsigned(9)
   78 18                                # text(24)
      74656D70657261747572655F67206669726D776172655F70
      # "temperature_g firmware_p"
   18 33                                # unsigned(51)
   44                                   # bytes(4)
      ABCD0000
   18 34                                # unsigned(52)
   41                                   # bytes(1)
      00
   08                                   # unsigned(8)
   A1                                   # map(1)
      0E                                # unsigned(14)
      A2                                # map(2)
         02                             # unsigned(2)
         77                             # text(23)
            34322D35302D33312D46462D45462D33372D33322D3339
            # "42-50-31-FF-EF-37-32-39"
         08                             # unsigned(8)
         A1                             # map(1)
            01                          # unsigned(1)
            A4                          # map(4)
               01                       # unsigned(1)
               02                       # unsigned(2)
               20                       # negative(0)
               01                       # unsigned(1)
               21                       # negative(1)
               58 20                    # bytes(32)
                  D7CC072DE2205BDC1537A543D53C60A6
                  ACB62ECCD890C7FA27C9E354089BBE13
               22                       # negative(2)
               58 20                    # bytes(32)
                  F95E1D4B851A2CC80FFF87D8E23F22AF
                  B725D535E515D020731E79A3B4E47120
```

Figure 6: Example CWT Claims Set Using CBOR Encoding

3.3.1. Update of Access Rights

[

TODO: Specify how the AS issues an access token that dynamically updates the access rights of C. (See below for pre-requirements and a high-level direction)

(This should be specified with content in the present section, as well as in Section 3.2 and Section 4.4).

At the moment, this profile does not support the dynamic update of access rights for the client like other transport profiles of ACE do.

This can be enabled by building on concepts defined in [I-D.ietf-ace-workflow-and-params]:

* "Token series" - In this profile, it would be specialized as a set of consecutive access tokens issued by the AS for the pair (C, AUD), where C is the client whose public authentication credential is bound to those access tokens, while AUD is the audience for which C requests those access tokens.

* "token_series_id" - At the time of writing, [I-D.ietf-ace-workflow-and-params] describes the intended direction for defining this new prospective parameter, to be used in the access token request/response exchange between C and the AS.

  This parameter is meant to specify the unique identifier of a token series. In parallel, it is planned to define a new, corresponding claim to include into access tokens.

At a high-level, the above can enable the dynamic update of access rights as follows:

* Each access token in a token series includes the claim "token_series_id", with value the identifier of the token series that the access token belongs to.

* When issuing the first access token in a token series, the AS includes the parameter "token_series_id" in the access token response to the client, with value the identifier of the token series that the access token belongs to.

* When C requests from the AS an access token that dynamically updates its current access rights to access protected resources at the same audience, C sends to the AS an access token request such that:

  - It includes the parameter "token_series_id", with value the identifier of the token series for which the new access token is requested.

  - It does _not_ include the parameters "context_id", "salt_input", and "client_cred_verify" or "client_cred_verify_mac".

* If the AS issues the new access token that dynamically updates the access rights of C, then the access token includes the claim "token_series_id", with value the identifier of the same token series for which the access token has been issued.

When receiving the new access token, the RS uses the value of the claim "token_series_id", and identifies the stored old access token that has to be superseded by the new one, as both belonging to the same token series.

]

3.3.2. 'context_id' Claim

The 'context_id' claim provides a value that the client requesting the access token wishes to use with the RS, as a hint for a security context.

This parameter specifies the value of the Context ID input, encoded as a CBOR byte string.

3.3.3. 'salt_input' Claim

The 'salt_input' claim provides a value that the client requesting the access token wishes to use as a part of a salt with the RS, e.g., for deriving cryptographic material.

This parameter specifies the value of the salt input, encoded as a CBOR byte string.

4. Client-RS Communication

This section details the POST request and response to the authz-info endpoint between the client and the RS.

The proof of possession required to bind the access token to the client is explicitly performed when the RS receives and verifies a request from the client protected with Group OSCORE, either with the group mode (see Section 7 of [I-D.ietf-core-oscore-groupcomm]) or with the pairwise mode (see Section 8 of [I-D.ietf-core-oscore-groupcomm]).

In particular, the RS uses the client's public key bound to the access token, either when verifying the signature of the request (if protected with the group mode), or when verifying the request as integrity-protected with pairwise keying material derived from the two peers' authentication credentials and asymmetric keys (if protected with the pairwise mode). In either case, the RS also authenticates the client.

Similarly, when receiving a protected response from the RS, the client uses the RS's public key either when verifying the signature of the response (if protected with the group mode), or when verifying the response as integrity-protected with pairwise keying material derived from the two peers' authentication credentials and asymmetric keys (if protected with the pairwise mode). In either case, the client also authenticates the RS. Mutual authentication is only achieved after the client has successfully verified the protected response from the RS.

Therefore, an attacker using a stolen access token cannot generate a valid Group OSCORE message as protected through the client's private key, and thus cannot prove possession of the PoP key bound to the access token. Also, if a client legitimately owns an access token but has not joined the OSCORE group, it cannot generate a valid Group OSCORE message, as it does not store the necessary keying material shared among the group members.

Furthermore, a client C1 is supposed to obtain a valid access token from the AS, as specifying its own authentication credential (and the public key included thereof) associated with its own private key used in the OSCORE group, together with its own Sender ID in that OSCORE group (see Section 3.2). This allows the RS receiving the access token to verify with the Group Manager of that OSCORE group whether such a client indeed has that Sender ID and uses that authentication credential in the OSCORE group.

As a consequence, a different client C2, also member of the same OSCORE group, is not able to impersonate C1, by: i) getting a valid access token, specifying the Sender ID of C1 and a different (made-up) authentication credential; ii) successfully uploading the access token to the RS; and then iii) attempting to communicate using Group OSCORE and impersonating C1, while also blaming C1 for the consequences of the interaction with the RS.

4.1. C-to-RS POST to authz-info Endpoint

The client uploads the access token to the authz-info endpoint of the RS, as defined in Section 5.10.1 of [RFC9200].

4.2. RS-to-C: 2.01 (Created)

The RS MUST verify the validity of the access token as defined in Section 5.10.1 of [RFC9200], with the following additions.

* The RS checks that the claims 'context_id', 'salt_input', and 'cnf' are included in the access token.

  If any of these claims are missing or malformed, the RS MUST consider the access token invalid and MUST reply to the client with an error response code equivalent to the CoAP code 4.00 (Bad Request).

  Otherwise, the RS retrieves:

  - GID* as the GID of the OSCORE group, which is specified in the 'context_id' claim.

  - SID* as the Sender ID that the client has in the OSCORE group, which is specified in the 'salt_input' claim.

  - AUTH_CRED_C* as the authentication credential that the client uses in the OSCORE group, which is specified in the inner confirmation value of the 'cnf' claim.

* The RS builds GROUPS as the set of OSCORE groups such that all the following conditions hold, for each group G in the set.

  - The RS is a member of the group G.

  - The group G has GID* as current GID.

  - The audience targeted by the access token is consistent with using the group G for accessing protected resources hosted by the RS.

If no such group is found, the RS MUST consider the access token invalid and MUST reply to the client with an error response code equivalent to the CoAP code 4.00 (Bad Request).

Otherwise, for each of the N >= 1 groups G in the set GROUPS, the RS MUST request to the corresponding Group Manager the authentication credential that the client uses in G, specifying SID* in the request sent to the Group Manager (see Section 9.3 of [I-D.ietf-ace-key-groupcomm-oscore]).

When receiving a successful response from each of the Group Managers, the RS MUST check whether the client's authentication credential AUTH_CRED_C retrieved from the Group Manager is equal to AUTH_CRED_C* retrieved from the access token. In case AUTH_CRED_C* is provided within a chain or a bag, but AUTH_CRED_C is not provided within the same chain or bag, then the RS MUST NOT determine AUTH_CRED_C* and AUTH_CRED_C to be equal.

If any of the following conditions hold, the RS MUST consider the access token invalid and MUST reply to the client with an error response code equivalent to the CoAP code 5.03 (Service Unavailable).

- None or more than one of the Group Managers provide the RS with a successful response where the conveyed AUTH_CRED_C is equal to AUTH_CRED_C*.

- After having performed a maximum, pre-configured amount of attempts or after a maximum, pre-configured amount of time has elapsed, less than N Group Managers have sent a successful response to the RS.

The process above is successful if and only if the RS receives a successful response from all the N Group Managers, and exactly one of such responses conveys AUTH_CRED_C equal to AUTH_CRED_C*. This ensures that there is only one OSCORE group G* such that: the client and the RS are both its members; it has GID* as current GID; and the client uses SID* as Sender ID in the group. In turn, this will ensure that the RS can bound the access token to such single OSCORE group G*.

If the operations above are successful, the access token is valid, and further checks on its content are successful, then the RS associates the authorization information from the access token with the Group OSCORE Security Context of the OSCORE group G*.

In particular, the RS associates the authorization information from the access token with the quartet (GID, SaltInput, AuthCred, AuthCredGM), where:

* GID is the Group Identifier of G* (i.e., GID*).

* SaltInput is the Sender ID that the client uses in G* (i.e., SID*).

* AuthCred is the authentication credential that the client uses in G* (i.e., AUTH_CRED_C*).

* AuthCredGM is the authentication credential of the Group Manager responsible for G* (see Section 2.1.6 of [I-D.ietf-core-oscore-groupcomm]).

The RS MUST keep this association up-to-date over time, as the quartet (GID, SaltInput, AuthCred, AuthCredGM) associated with the access token might change. In particular:

* If the OSCORE group is rekeyed (see Section 12.2 of [I-D.ietf-core-oscore-groupcomm] and Section 11 of [I-D.ietf-ace-key-groupcomm-oscore]), the Group Identifier also changes in the group and the new one replaces the current 'GID' value in the quartet (GID, SaltInput, AuthCred, AuthCredGM).

* If the client requests and obtains a new OSCORE Sender ID from the Group Manager (see Section 2.6.3.1 of [I-D.ietf-core-oscore-groupcomm] and Section 9.2 of [I-D.ietf-ace-key-groupcomm-oscore]), the new Sender ID replaces the current 'SaltInput' value in the quartet (GID, SaltInput, AuthCred, AuthCredGM).

  Among the quartets corresponding to access tokens that are associated with a given Group OSCORE Security Context, the RS can always identify the correct quartet to update. For example, the RS can leverage the triple (GID, AuthCred, AuthCredGM) that played a role in the successful decryption and verification of a request protected with Group OSCORE and sent by the client with the newly obtained Sender ID.

* If the Group Manager of the OSCORE group changes its authentication credential, the new authentication credential of the Group Manager replaces the current 'AuthCredGM' value in the quartet (GID, SaltInput, AuthCred, AuthCredGM).

In order to obtain the latest authentication credential of the Group Manager, the RS can re-join the group (see Section 6 of [I-D.ietf-ace-key-groupcomm-oscore]) or send a dedicated request to the Group Manager (see Section 9.3 of [I-D.ietf-ace-key-groupcomm-oscore]).

As defined in Section 5, a possible change of the client's authentication credential requires the client to upload to the RS a new access token bound to the new authentication credential.

Finally, the RS MUST send a 2.01 (Created) response to the client, as defined in Section 5.10.1 of [RFC9200].

4.3. Client-RS Secure Communication

When previously joining the OSCORE group, both the client and the RS have already established the related Group OSCORE Security Context to communicate as group members. Therefore, they can simply start to securely communicate using Group OSCORE, without deriving any additional keying material or security association.

If either of the client or the RS deletes an access token (e.g., when the access token has expired or has been revoked), it MUST NOT delete the related Group OSCORE Security Context. The client MAY request a new access token from the AS, to be uploaded to the RS for re-enabling access to protected resources for the client.

4.3.1. Client Side

After having received the 2.01 (Created) response from the RS, following the POST request to the authz-info endpoint, the client starts the communication with the RS, by sending a request protected with Group OSCORE using the Group OSCORE Security Context [I-D.ietf-core-oscore-groupcomm].

When communicating with the RS to access the resources as specified by the authorization information, the client MUST use the Group OSCORE Security Context of the pertinent OSCORE group, whose GID was specified in the 'context_id' parameter of the access token request.

4.3.2. Resource Server Side

After successful validation of the access token as defined in Section 4.2 and after having sent the 2.01 (Created) response, the RS can start to communicate with the client using Group OSCORE [I-D.ietf-core-oscore-groupcomm].

When processing an incoming request protected with Group OSCORE, the RS MUST consider as valid client's authentication credential only the one associated with the stored access token. As defined in Section 5, a possible change of the client's authentication credential requires the client to upload to the RS a new access token bound to the new authentication credential.

For every incoming request, if Group OSCORE verification succeeds, the verification of access rights is performed as described in Section 4.5.

If the RS receives a request protected with a Group OSCORE Security Context CTX, the target resource requires authorization, and the RS does not store a valid access token related to CTX, then the RS MUST reply with a 4.01 (Unauthorized) error response protected with CTX.

4.4. Update of Access Rights

[

TODO: Specify the processing on the RS when receiving an access token that dynamically updates the access rights of C. (See Section 3.3.1 for pre-requirements and a high-level direction)

]

4.5. Access Rights Verification

The RS MUST follow the procedures defined in Section 5.10.2 of [RFC9200]. If an RS receives a request protected with Group OSCORE from a client, the RS processes the request according to [I-D.ietf-core-oscore-groupcomm].

If the Group OSCORE verification succeeds and the target resource requires authorization, the RS retrieves the authorization information from the access token associated with the Group OSCORE Security Context.

In particular, the RS retrieves the access token associated with the quartet (GID, SaltInput, AuthCred, AuthCredGM), where:

* GID is the Group Identifier of the OSCORE group, which is specified in the 'kid context' parameter of the CoAP OSCORE option in the received request. The RS maintains GID in its Common Context within the Group OSCORE Security Context associated with the OSCORE group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]).

* SaltInput is the Sender ID that the client has in the OSCORE group, which is specified in the 'kid' parameter of the CoAP OSCORE option in the received request. The RS maintains SaltInput in its Recipient Context associated with the client, within the Group OSCORE Security Context associated with the OSCORE group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]).

* AuthCred is the authentication credential that the client uses in the OSCORE group. The RS maintains AuthCred in its Recipient Context associated with the client, within the Group OSCORE Security Context associated with the OSCORE group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]).

* AuthCredGM is the authentication credential of the Group Manager responsible for the OSCORE group. The RS maintains AuthCredGM in its Common Context within the Group OSCORE Security Context associated with the OSCORE group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]).

Then, the RS MUST verify that the action requested on the resource is authorized.

If the RS has no valid access token for the client, the RS MUST reject the request and MUST reply to the client with a 4.01 (Unauthorized) error response.

If the RS has an access token for the client but no actions are authorized on the target resource, the RS MUST reject the request and MUST reply to the client with a 4.03 (Forbidden) error response.

If the RS has an access token for the client but the requested action is not authorized, the RS MUST reject the request and MUST reply to the client with a 4.05 (Method Not Allowed) error response.

4.6. Storing Multiple Access Tokens per PoP Key

According to Section 5.10.1 of [RFC9200], an RS is recommended to store only one access token per proof-of-possession (PoP) key, and to supersede such an access token when receiving and successfully validating a new one bound to the same PoP key.

However, when using the profile specified in this document, an RS might practically have to deviate from that recommendation and store multiple access tokens bound to the same PoP key, i.e., to the same public authentication credential of a client.

For example, this can occur in the following cases.

* The RS is the single RS associated with an audience AUD1 and also belongs to a group-audience AUD2 (see Section 6.9 of [RFC9200]).

  A client C with public authentication credential AUTH_CRED_C can request two access tokens T1 and T2 from the AS, such that:

  - T1 targets AUD1 and has scope SCOPE1; and

  - T2 targets AUD2 and has scope SCOPE2 different from SCOPE1.

  Both T1 and T2 are going to be bound to the same PoP key specified by AUTH_CRED_C.

  In fact, if the AS issues access tokens targeting a group-audience, then the above can possibly be the case when using any transport profile of ACE that supports asymmetric PoP keys. If so, the RS should be ready to store at minimum one access token per PoP key per audience it belongs to.

* The RS is a member of two OSCORE groups G1 and G2. In particular, the same format of public authentication credentials is used in both OSCORE groups.

  A client C with public authentication credential AUTH_CRED_C of such format, also member of the two OSCORE groups G1 and G2, can conveniently use AUTH_CRED_C as its public authentication credential in both those groups. Therefore, C can request two access tokens T1 and T2 from the AS, such that:

  - T1 targets RS and reflects the membership of C in G1, as per its claims "context_id" and "salt_input"; and

  - T2 targets RS and reflects the membership of C in G2, as per its claims "context_id" and "salt_input".

  Both T1 and T2 are going to be bound to the same PoP key specified by AUTH_CRED_C.

  When using the profile specified in this document, the RS should be ready to store at minimum one access token per PoP key per OSCORE group it is a member of (although, per the previous point, even this can still be limiting).

* The RS uses both the profile specified in this document and a different transport profile of ACE that also relies on asymmetric PoP keys, e.g., the EDHOC and OSCORE profile defined in [I-D.ietf-ace-edhoc-oscore-profile].

In such a case, a client C with public authentication credential AUTH_CRED_C can request two access tokens T1 and T2 from the AS, such that:

- T1 targets RS and is meant to be used according to the Group OSCORE profile defined in this document; and

- T2 targets RS and is meant to be used according to the EDHOC and OSCORE profile defined in [I-D.ietf-ace-edhoc-oscore-profile].

Both T1 and T2 are going to be bound to the same PoP key specified by AUTH_CRED_C.

When using multiple transport profiles of ACE that rely on asymmetric PoP keys, it is reasonable that the RS is capable to store at minimum one access token per PoP key per used profile (although, per the previous points, even this can still be limiting).

In order to keep the same spirit of the recommendation in Section 5.10.1 of [RFC9200] without impeding cases such as those outlined in the examples above, the following defines a more fine-grained recommendations for the storage of access tokens at an RS when this profile is used.

* As to access tokens issued in accordance with this profile (i.e., specifying the profile "coap_group_oscore"), it is RECOMMENDED that an RS stores only one access token that:

  - is bound to a specific PoP key;

  - targets a specific audience; and

  - is related to a specific OSCORE group.

* As to access tokens issued in accordance with a different profile P that an RS may use in parallel with the profile defined in this document, it is RECOMMENDED that an RS stores only one access token that:

  - is issued in accordance with the profile P;

  - is bound to a specific PoP key;

  - targets a specific audience; and

- is related to a specific secure association used by the client to protect communications with the RS.

In accordance with these recommendations, if an access token T_NEW does not differ in any of the respects above from an existing access token T_OLD stored at the RS, then T_NEW ought to supersede T_OLD by replacing the corresponding authorization information.

Not complying with these recommendations can additionally complicate (constrained) implementations of RSs, with respect to required storage and the validation of a protected request against the applicable, stored access tokens associated with the same client. That is, it increases the strain on an RS in determining the actual permissions of a client, e.g., if access tokens contradict each other and thus might lead the RS to enforce wrong permissions. Moreover, if one of the access tokens expires earlier than others, the resulting permissions may offer insufficient protection.

5. Change of Client's Authentication Credential in the Group

During its membership in the OSCORE group, the client might change the authentication credential that it uses in the group. When this happens, the client uploads the new authentication credential to the Group Manager, as defined in Section 9.4 of [I-D.ietf-ace-key-groupcomm-oscore].

After that, and in order to continue communicating with the RS, the client MUST perform the following actions.

1. The client requests a new access token T_NEW to the AS, as defined in Section 3. In particular, when sending the access token request as defined in Section 3.2, the client specifies:

   * The current Group Identifier of the OSCORE group, as the value of the 'context_id' parameter.

   * The current Sender ID that it has in the OSCORE group, as the value of the 'salt_input' parameter.

   * The new authentication credential that it uses in the OSCORE group, as the inner confirmation value of the 'req_cnf' parameter.

   * Optionally, the proof-of-possession (PoP) evidence corresponding to the public key of the new authentication credential, as the value of the 'client_cred_verify' or 'client_cred_verify_mac' parameter.

The possible omission of the PoP evidence is based on the same criteria that are defined in Section 3.2.

2. After receiving the access token response from the AS (see Section 3.3), the client performs with the RS the same exchanges that are defined in Section 4.

When receiving the new access token T_NEW, the RS performs the same steps defined in Section 4.2, with the following addition in case the new access token is successfully verified and stored:

* The RS deletes any stored access token T_OLD such that the associated quartet (GID, SaltInput, AuthCred, AuthCredGM) differs from the same quartet associated with T_NEW only as to the value of AuthCred.

6. Secure Communication with the AS

As specified in the ACE framework (see Sections 5.8 and 5.9 of [RFC9200]), the requesting entity (client and/or RS) and the AS communicate via the token or introspect endpoint. The use of CoAP and OSCORE [RFC8613] for this communication is RECOMMENDED in this profile. Other protocols fulfilling the security requirements defined in Sections 5 and 6 of [RFC9200] (such as HTTP and DTLS or TLS) MAY be used instead.

If OSCORE [RFC8613] is used, the requesting entity and the AS are expected to have a pre-established Security Context in place. How this Security Context is established is out of the scope of this profile. Furthermore, the requesting entity and the AS communicate using OSCORE through the token endpoint as specified in Section 5.8 of [RFC9200] and through the introspect endpoint as specified in Section 5.9 of [RFC9200].

7. Discarding the Security Context

As members of an OSCORE group, the client and the RS may independently leave the group or be forced to, e.g., if compromised or suspected so. Upon leaving the OSCORE group, the client or RS also discards the Group OSCORE Security Context, which may anyway be renewed by the Group Manager through a group rekeying process (see Section 12.2 of [I-D.ietf-core-oscore-groupcomm]).

The client or RS can acquire a new Group OSCORE Security Context by re-joining the OSCORE group, e.g., by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore]. In such a case, the client SHOULD request a new access token to be uploaded to the RS.

8. Guidelines on Using Multiple Profiles

When using the profile defined in this document, access tokens are to be bound to a Group OSCORE Security Context, which is used to protect communications between the client and the RS(s) in the targeted audience by using the security protocol Group OSCORE.

After having obtained an access token T1 for this profile and uploaded it to the RS (RSs) pertaining to the targeted audience, the client might want to establish a separate, pairwise OSCORE association with that RS (with one RS among those RSs). In order to do that, the client can ask the AS for a different access token T2 intended for that RS (for one RS among those RSs), as per the OSCORE profile defined in [RFC9203].

Since the ACE framework does not allow the client to negotiate with the AS the profile to use, the client has instead to choose the use of the OSCORE profile, and to explicitly indicate it to the AS when requesting T2.

To this end, the client may indicate its wish for an access token aligned with the Group OSCORE profile or with the OSCORE profile, by specifying one of two different audiences in the ’audience’ parameter of the access token request to the AS. Assuming a proper configuration of the access policies at the AS, this is still conducive to a consistent evaluation of what is specified in the ’scope’ parameter of the access token request against the access policies at the AS.

For example, an RS registered as "rs1" at the AS can be associated with two audiences:

* "rs1_gp_osc", which the client can use to request an access token for the Group OSCORE profile and targeting (also) that RS. That is, the client specifies this audience when requesting the access token T1.

* "rs1_osc", which the client can use to request an access token for the OSCORE profile and targeting only that RS. That is, the client specifies this audience when requesting the access token T2.

Editor’s note: the approach above is provisional and intended to simply be an informative example. In future versions of this Internet Draft, such an approach might be complemented or replaced by the use of the ’ace_profile’ parameter, as included in the access token request per its extended semantics defined in [I-D.ietf-ace-workflow-and-params].

Note that an RS has to be able to store at least one access token per PoP key. When specifically considering the Group OSCORE profile and the OSCORE profile, the RS can always store both corresponding access tokens T1 and T2, since they are always bound to different PoP keys. That is:

* In the Group OSCORE profile, the PoP key is the client's public key, which is included in the client's authentication credential specified in the 'cnf' claim of the access token.

* In the OSCORE profile, the PoP key is fundamentally an OSCORE Master Secret, which is specified within the OSCORE_Input_Material object of the 'cnf' claim of the access token.

The same approach discussed above can be used in case the profile used for the access token T2 is instead the EDHOC and OSCORE profile defined in [I-D.ietf-ace-edhoc-oscore-profile]. In such a case, the same PoP key might be bound to both T1 and T2, i.e., if the client's public key is included both in the authentication credential that the client uses in the OSCORE group and in the authentication credential that the client uses as CRED_I (CRED_R) when running the EDHOC protocol in the forward (reverse) message flow (see Appendix A.2 of [RFC9528]).

Section 4.6 provides considerations and recommendations on storing multiple access tokens per PoP key when using the Group OSCORE profile, also in parallel with alternative profiles.

9. CBOR Mappings

The new parameters defined in this document MUST be mapped to CBOR types as specified in Table 1, using the given integer abbreviation for the map key.

Parameter name	CBOR Key	Value Type
context_id	TBD	byte string
salt_input	TBD	byte string
client_cred_verify	TBD	byte string
client_cred_verify_mac	TBD	byte string

Table 1: CBOR Mappings for New Parameters

The new claims defined in this document MUST be mapped to CBOR types as specified in Table 2, using the given integer abbreviation for the map key.

Claim name	CBOR Key	Value Type
context_id	TBD	byte string
salt_input	TBD	byte string

Table 2: CBOR Mappings for New Claims

10. Security Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [RFC9200]. Thus, the general security considerations from the ACE framework also apply to this profile.

The proof-of-possession (PoP) key bound to an access token is always an asymmetric key, i.e., the public key included in the authentication credential that the client uses in the OSCORE group. This means that there is never a same shared secret used as PoP key with possible multiple RSs. Therefore, it is possible and safe for the AS to issue an access token for an audience that includes multiple RSs (i.e., a group-audience, see Section 6.9 of [RFC9200]).

In such a case, as per Section 6.1 of [RFC9200], the AS has to ensure the integrity protection of the access token by protecting it through an asymmetric signature. In addition, the used group-audience has to correctly identify all the RSs that are intended recipients of the access token and for which the single scope specified in the access token applies. As a particular case, the audience can refer to the OSCORE group as a whole, if the access token is intended for all the RSs in that group.

Furthermore, this document inherits the general security considerations about Group OSCORE [I-D.ietf-core-oscore-groupcomm], as to the specific use of Group OSCORE according to this profile.

Group OSCORE is designed to secure point-to-point as well as point-to-multipoint communications, providing a secure binding between a single request and multiple corresponding responses. In particular, Group OSCORE fulfills the same security requirements of OSCORE.

Group OSCORE ensures source authentication of messages both in group mode (see Section 7 of [I-D.ietf-core-oscore-groupcomm]) and in pairwise mode (see Section 8 of [I-D.ietf-core-oscore-groupcomm]).

When protecting an outgoing message in group mode, the sender uses its private key to compute a digital signature that is embedded in the protected message. The group mode can be used to protect messages sent to multiple recipients (e.g., over IP multicast) or to a single recipient.

When protecting an outgoing message in pairwise mode, the sender uses a pairwise symmetric key that is derived from the asymmetric keys of the two peers exchanging the message. The pairwise mode can be used to protect only messages intended for a single recipient.

11. Privacy Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [RFC9200]. Thus the general privacy considerations from the ACE framework also apply to this profile.

As this profile uses Group OSCORE, the privacy considerations from [I-D.ietf-core-oscore-groupcomm] apply to this document as well.

An unprotected response to an unauthorized request may disclose information about the RS and/or its existing relationship with the client. It is advisable to include as little information as possible in an unencrypted response. However, since both the client and the RS share a Group OSCORE Security Context, unauthorized yet protected requests are followed by protected responses, which can thus include more detailed information.

Although it may be encrypted, the access token is sent in the clear to the authz-info endpoint at the RS. Thus, if the client uses the same single access token from multiple locations with multiple resource servers, it can risk being tracked through the access token’s value.

Note that, even though communications are protected with Group OSCORE, some information might still leak, due to the observable size, source address, and destination address of exchanged messages.

12. IANA Considerations

This document has the following actions for IANA.

Note to RFC Editor: Please replace "[RFC-XXXX]" with the RFC number of this document and delete this paragraph.

12.1. ACE Profiles Registry

IANA is asked to add the following entry to the "ACE Profiles" registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group, following the procedure specified in Section 8.8 of [RFC9200].

* Name: coap_group_oscore

* Description: Profile to secure communications between constrained nodes using the Authentication and Authorization for Constrained Environments framework, by enabling authentication and fine-grained authorization of members of an OSCORE group that use a pre-established Group OSCORE Security Context to communicate with Group OSCORE.

* CBOR Value: TBD (value between 1 and 23)

* Reference: [RFC-XXXX]

12.2. OAuth Parameters Registry

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

* Name: context_id

* Parameter Usage Location: token request

* Change Controller: IETF

* Reference: Section 3.2.1 of [RFC-XXXX]

* Name: salt_input

* Parameter Usage Location: token request

* Change Controller: IETF

* Reference: Section 3.2.2 of [RFC-XXXX]

* Name: client_cred_verify

* Parameter Usage Location: token request

* Change Controller: IETF

* Reference: Section 3.2.3 of [RFC-XXXX]

* Name: client_cred_verify_mac

* Parameter Usage Location: token request

* Change Controller: IETF

* Reference: Section 3.2.4 of [RFC-XXXX]

12.3. OAuth Parameters CBOR Mappings Registry

IANA is asked to add the following entries to the "OAuth Parameters CBOR Mappings" registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group, following the procedure specified in Section 8.10 of [RFC9200].

* Name: context_id

* CBOR Key: TBD (value between 1 and 255)

* Value Type: byte string

* Reference: Section 3.2.1 of [RFC-XXXX]

* Original Specification: [RFC-XXXX]

* Name: salt_input

* CBOR Key: TBD (value between 1 and 255)

* Value Type: byte string

* Reference: Section 3.2.2 of [RFC-XXXX]

* Original Specification: [RFC-XXXX]

* Name: client_cred_verify

* CBOR Key: TBD (value between 1 and 255)

* Value Type: byte string

* Reference: Section 3.2.3 of [RFC-XXXX]

* Original Specification: [RFC-XXXX]

* Name: client_cred_verify_mac

* CBOR Key: TBD (value between 1 and 255)

* Value Type: byte string

* Reference: Section 3.2.4 of [RFC-XXXX]

* Original Specification: [RFC-XXXX]

12.4. CBOR Web Token (CWT) Claims Registry

IANA is asked to add the following entries to the "CBOR Web Token (CWT) Claims" registry within the "CBOR Web Token (CWT) Claims" registry group, following the procedure specified in Section 9.1 of [RFC8392].

* Claim Name: context_id

* Claim Description: Client provided Context ID

* JWT Claim Name: N/A

* Claim Key: TBD (value between 1 and 255)

* Claim Value Type: byte string

* Change Controller: IETF

* Reference: Section 3.3.2 of [RFC-XXXX]

* Claim Name: salt_input

* Claim Description: Client provided salt input

* JWT Claim Name: N/A

* Claim Key: TBD (value between 1 and 255)

* Claim Value Type: byte string

* Change Controller: IETF

* Reference: Section 3.3.3 of [RFC-XXXX]

12.5. TLS Exporter Label Registry

IANA is asked to add the following entry to the "TLS Exporter Label" registry within the "Transport Layer Security (TLS) Parameters" registry group, following the procedure specified in Section 6 of [RFC5705] and updated in Section 12 of [RFC8447].

* Value: EXPORTER-ACE-PoP-Input-Client-AS

* DTLS-OK: Y

* Recommended: N

* Reference: Section 3.2 of [RFC-XXXX]

13. References

13.1. Normative References

[I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M. and F. Palombini, "Key Management for Group Object Security for Constrained RESTful Environments (Group OSCORE) Using Authentication and Authorization for Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-oscore-18, 28 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-oscore-18>.

[I-D.ietf-core-groupcomm-bis]
Dijk, E. and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-14, 2 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-14>.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and R. H旦glund, "Group Object Security for Constrained

RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-26, 5 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-26>.

[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/rfc/rfc2119>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/rfc/rfc5246>.

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

[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/rfc/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/rfc/rfc6347>.

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

[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/rfc/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/rfc/rfc7748>.

[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/rfc/rfc8174>.

[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/rfc/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/rfc/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/rfc/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/rfc/rfc8613>.

[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/rfc/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/rfc/rfc8949>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

[RFC9200] Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments Using the OAuth 2.0 Framework (ACE-OAuth)", RFC 9200, DOI 10.17487/RFC9200, August 2022, <https://www.rfc-editor.org/rfc/rfc9200>.

[RFC9201] Seitz, L., "Additional OAuth Parameters for Authentication and Authorization for Constrained Environments (ACE)", RFC 9201, DOI 10.17487/RFC9201, August 2022, <https://www.rfc-editor.org/rfc/rfc9201>.

[RFC9203] Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "The Object Security for Constrained RESTful Environments (OSCORE) Profile of the Authentication and Authorization for Constrained Environments (ACE) Framework", RFC 9203, DOI 10.17487/RFC9203, August 2022, <https://www.rfc-editor.org/rfc/rfc9203>.

13.2. Informative References

[I-D.ietf-ace-edhoc-oscore-profile]
Selander, G., Mattsson, J. P., Tiloca, M., and R. H旦glund, "Ephemeral Diffie-Hellman Over COSE (EDHOC) and Object Security for Constrained Environments (OSCORE) Profile for Authentication and Authorization for Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-edhoc-oscore-profile-08, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-edhoc-oscore-profile-08>.

[I-D.ietf-ace-workflow-and-params]
Tiloca, M. and G. Selander, "Short Distribution Chain (SDC) Workflow and New OAuth Parameters for the Authentication and Authorization for Constrained Environments (ACE) Framework", Work in Progress, Internet-Draft, draft-ietf-ace-workflow-and-params-05, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-workflow-and-params-05>.

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., H旦glund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-15, 18 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-15>.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Ams端ss, C., and P. Van der Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-discovery-17, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-tiloca-core-oscore-discovery-17>.

[NIST-800-207]
Rose, S., Borchert, O., Mitchell, S., and S. Connelly, "Zero Trust Architecture - NIST Special Publication 800-207", August 2020, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[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/rfc/rfc8446>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[RFC9202] 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)", RFC 9202, DOI 10.17487/RFC9202, August 2022, <https://www.rfc-editor.org/rfc/rfc9202>.

[RFC9431] Sengul, C. and A. Kirby, "Message Queuing Telemetry Transport (MQTT) and Transport Layer Security (TLS) Profile of Authentication and Authorization for Constrained Environments (ACE) Framework", RFC 9431, DOI 10.17487/RFC9431, July 2023, <https://www.rfc-editor.org/rfc/rfc9431>.

[RFC9528] Selander, G., Preu Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, DOI 10.17487/RFC9528, March 2024, <https://www.rfc-editor.org/rfc/rfc9528>.

Appendix A. Profile Requirements

This appendix lists the specifications of this profile based on the requirements of the ACE framework, as requested in Appendix C of [RFC9200].

* Optionally, define new methods for the client to discover the necessary permissions and AS for accessing a resource, different from the one proposed in [RFC9200]: Not specified.

* Optionally, specify new grant types: Not specified.

* Optionally, define the use of client certificates as client credential type: Not specified.

* Specify the communication protocol the client and RS must use: CoAP.

* Specify the security protocol the client and RS must use to protect their communication: Group OSCORE, by using a pre-established Group OSCORE Security Context.

* Specify how the client and the RS mutually authenticate: Explicitly, by possession of a common Group OSCORE Security Context and by either: usage of digital signatures embedded in messages, if protected with the group mode of Group OSCORE; or protection of messages with the pairwise mode of Group OSCORE, by using pairwise symmetric keys derived from the asymmetric keys of the two peers exchanging the message. Note that mutual authentication is not completed before the client has verified a Group OSCORE response using the corresponding Group OSCORE Security Context.

* Specify the proof-of-possession protocol(s) and how to select one, if several are available. Also specify which key types (e.g., symmetric/asymmetric) are supported by a specific proof-of-possession protocol: Group OSCORE algorithms; asymmetric keys verified and distributed by a Group Manager.

* Specify a unique ace_profile identifier: coap_group_oscore.

* If introspection is supported, specify the communication and security protocol for introspection: HTTP/CoAP (+ TLS/DTLS/OSCORE).

* Specify the communication and security protocol for interactions between the client and AS: HTTP/CoAP (+ TLS/DTLS/OSCORE).

* Specify if/how the authz-info endpoint is protected, including how error responses are protected: Not protected.

* Optionally, define other methods of token transport than the authz-info endpoint: Not defined.

Appendix B. CDDL Model

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

```
; ACE Profiles
coap_group_oscore = 5

; OAuth Parameters CBOR Mappings
context_id_param = 71
salt_input_param = 72
client_cred_verify = 73
client_cred_verify_mac = 74

; CBOR Web Token (CWT) Claims
context_id_claim = 51
salt_input_claim = 52

; CWT Confirmation Methods
kccs = 11
```

Figure 7: CDDL model

Appendix C. Document Updates

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

C.1. Version -04 to -05

* Consistent mentioning of the optional presence of the PoP evidence in the access token request.

* Guidelines on updating the quartet (GID, SaltInput, AuthCred, AuthCredGM) at the RS when a client obtains a new Sender ID.

* Clarifications and editorial improvements.

C.2. Version -03 to -04

* Required that ’cnf’ in the access token includes exactly what C uploaded to the Group Manager.

* Made the PoP evidence in the access token request optional.

* Better example value for audience, when indicating the profile to use.

* Placeholder: possible use of the ’ace_profile’ parameter with extended semantics, for C to select the right profile.

* Suggested value ranges for codepoints to register.

* Aligned CBOR abbreviations to those used in other documents.

* Editorial fixes and improvements.

C.3. Version -02 to -03

* Lowercase "client", "resource server", "authorization server", and "access token".

* Consistent update of section numbers for external references.

* Mentioned that this profile can also use the ACE alternative workflow.

* Clarified that the client may ask for a new access token after the old one becomes invalid.

* Enforced uniqueness pre-requirements on the client’s group membership before requesting an access token.

* Added checks on uniqueness of clients’ group membership at the RS.

* Clarified the process of access right verification.

* Added fine-grained recommendations on storing multiple access tokens bound to the same PoP key.

* Added guidelines on using multiple profiles.

* Fixes in the IANA considerations.

* Editorial fixes and improvements.

C.4. Version -01 to -02

* CBOR diagnostic notation uses placeholders from a CDDL model.

* Renamed the claim ’contextId_input’ to ’context_id’.

* Revised examples.

* Placeholders and early direction for dynamic update of access rights.

* Added text on storing multiple access tokens per PoP key on the RS.

* Fixes in the IANA considerations.

* Editorial fixes and improvements.

C.5. Version -00 to -01

* Deleting an access token does not delete the Group OSCORE Security Context.

* Distinct computation of the PoP input when C and the AS use (D)TLS 1.2 or 1.3.

* Revised computation of the PoP input when C and the AS use OSCORE.

* Revised computation of the PoP evidence when the OSCORE group is a pairwise-only group.

* Clarified requirements on the AS for verifying the PoP evidence.

* Renamed the TLS Exporter Label for computing the PoP input.

* Editorial fixes and improvements.

Acknowledgments

Ludwig Seitz contributed as a co-author of initial versions of this document.

The authors sincerely thank Christian Amsss, Tim Hollebeek, Benjamin Kaduk, John Preu Mattsson, Dave Robin, Jim Schaad, and Gran Selander for their comments and feedback.

The work on this document has been partly supported by the Sweden’s Innovation Agency VINNOVA and the Celtic-Next projects CRITISEC and CYPRESS; and by the H2020 project SIFIS-Home (Grant agreement 952652).

Authors’ Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-164 40 Kista Kista
Sweden
Email: marco.tiloca@ri.se

Rikard Hglund
RISE AB
Isafjordsgatan 22
SE-164 40 Kista Kista
Sweden
Email: rikard.hoglund@ri.se

Francesca Palombini
Ericsson AB
Torshamnsgatan 23
SE-164 40 Kista Kista
Sweden
Email: francesca.palombini@ericsson.com