ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2020

M. Tiloca
R. Hoeglund
L. Seitz
RISE AB
F. Palombini
Ericsson AB

November 04, 2019

Group OSCORE Profile of the Authentication and Authorization for Constrained Environments Framework
draft-tiloca-ace-group-oscore-profile-01

Abstract

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework. The profile uses Object Security for Constrained RESTful Environments (OSCORE) and/or Group OSCORE to provide communication security between a Client and (a group of) Resource Server(s). Furthermore, the profile uses (Group) OSCORE to provide server authentication, and OSCORE to achieve proof-of-possession for a key owned by the Client and bound to an OAuth 2.0 Access Token. Also, the profile provides proof-of-group-membership for the Client, by securely binding the pre-established Group OSCORE Security Context to the pairwise OSCORE Security Context newly established with the Resource Server.

Status of This Memo

Table of Contents

1. Introduction

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

For some instances of such applications, it may be just fine to enforce access control in a straightforward and plain fashion. That is, it is assumed that any Client authorized to join the group and to get the group key material, is also implicitly authorized as a group member to perform any action at any resource of any Server in the group. An example of an application where such implicit authorization might be used is a 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 their status.

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, a more fine-grained authorization 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; whereas 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. In some settings, 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. For example, in a group of lightbulbs, typically none of them should be able to switch on and off other lightbulbs in the group.

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 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 do so. Besides, it would prevent a misconfigured or compromised device from initiating a high-priority command and lock out normal control.

Hence, in the cases discussed above, being a legitimate group member and having obtained the group key material is not supposed to imply any particular access rights. Thus, a more fine-grained access control model has to be enforced, e.g. by using the Authentication and Authorization for Constrained Environments (ACE) framework [I-D.ietf-ace-oauth-authz]. That is, a Client has to first obtain authorization credentials in the form of an Access Token, and post it to the Resource Server(s) in the group before accessing the intended resources.

The ACE framework delegates to separate profile documents how to secure communications between the Client and the Resource Server. However each of the current profiles of ACE defined in [I-D.ietf-ace-oscore-profile] [I-D.ietf-ace-dtls-authorize] [I-D.ietf-ace-mqtt-tls-profile] admits a single security protocol that cannot be used to protect group messages sent over IP multicast.

This document specifies a profile of ACE, where a Client uses CoAP [RFC7252] to communicate to a single Resource Server, or CoAP over IP multicast [RFC7390][I-D.dijk-core-groupcomm-bis] to communicate to multiple Resource Servers that are members of a group and share a common set of resources. This profile uses two complementary security protocols to provide secure communication between the Client and the Resource Server(s).

That is, this document defines the use of either Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] or Group OSCORE [I-D.ietf-core-oscore-groupcomm] to protect unicast requests addressed to a single Resource Server, as well as possible responses. Additionally, it defines the use of Group OSCORE to protect multicast

requests sent to a group of Resource Servers, as well as possible individual responses. The Client and the Resource Servers need to have already joined an OSCORE group, for instance by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore], which is also based on ACE.

The Client authorizes its access to the Resource Server by using an Access Token, which is bound to a key (the proof-of-possession key). This profile uses OSCORE to achieve proof of possession, and OSCORE or Group OSCORE to achieve server authentication. Furthermore, this profile provides proof of Client's membership to the correct OSCORE group, by securely binding the pre-established Group OSCORE Security Context to the pairwise OSCORE Security Context newly established between the Client and the Resource Server.

OSCORE specifies how to use CBOR Object Signing and Encryption (COSE) [RFC8152] to secure CoAP messages. Group OSCORE builds on OSCORE to support group communication, and ensures source authentication by means of digital countersignatures embedded in protected messages.

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 the CoAP protocol [RFC7252], as well as related to the protection and processing of CoAP messages through OSCORE [RFC8613], also in group communication scenarios through Group OSCORE [I-D.ietf-core-oscore-groupcomm]. These include the concept of Group Manager, as the entity responsible for a set of groups where communications among members are secured with Group OSCORE.

This document also refers to "pairwise OSCORE Security Context", i.e. an OSCORE Security Context established between only one Client and one Resource Server, and used to communicate with OSCORE [RFC8613].

Readers are expected to be familiar with the terms and concepts described in the ACE framework for authentication and authorization [I-D.ietf-ace-oauth-authz], as well as in the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile]. 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, unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol".

2. Protocol Overview

This section provides an overview on how to use the ACE framework for authentication and authorization [I-D.ietf-ace-oauth-authz] to secure communications between a Client and a (set of) Resource Server(s) using OSCORE [RFC8613] and/or Group OSCORE [I-D.ietf-core-oscore-groupcomm].

An overview of the protocol flow for this profile is shown in Figure 1. In the figure, it is assumed that C, RS1 and RS2 have previously joined an OSCORE group with Group Identifier (gid) "abcd0000", and got assigned Sender ID (sid) "0", "1" and "2" in the group, respectively. It is also assumed that both RS1 and RS2 are associated with the same AS. For simplicity, the figure does not show the preliminary phase where C, R1 and R2 join the OSCORE group.

```
C                            RS1           RS2                          AS
| [--- Resource Request --->] |             |                            |
|                             |             |                            |
| [<--- AS Information -----] |             |                            |
|                             |             |                            |
|-------- POST /token -------------------------------------------------->|
|  (aud: RS1, sid: 0, gid: abcd0000, ... )  |                            |
|                             |             |                            |
|<---------------------------------- Access Token + RS Information ------|
|                             |  (aud: RS1, sid: 0, gid: abcd0000, ... ) |
|---- POST /authz-info ------>|             |                            |
|    (access_token, N1)       |             |                            |
|                             |             |                            |
|<--- 2.01 Created (N2) ------|             |                            |
|                             |             |                            |
/Pairwise OSCORE Sec  /Pairwise OSCORE Sec  |                            |
 Context Derivation/   Context Derivation/  |                            |
|                             |             |                            |
|-------- POST /token -------------------------------------------------->|
|  (aud: RS2, sid: 0, gid: abcd0000, ... )  |                            |
|                             |             |                            |
|<---------------------------------- Access Token + RS Information ------|
|                             |  (aud: RS2, sid: 0, gid: abcd0000, ... ) |
|                             |             |                            |
|----- POST /authz-info ------------------->|                            |
|     (access_token, N1')      |            |                            |
|                              |            |                            |
|<--- 2.01 Created (N2') -------------------|                            |
|                              |            |                            |
/Pairwise OSCORE Sec           |  /Pairwise OSCORE Sec                   |
 Context Derivation/           |   Context Derivation/                   |
|                              |            |                            |
|------ OSCORE Request ------->|            |                            |
|    ?(abcd0000, N1, N2)       |            |                            |
|                              |            |                            |
|<----- OSCORE Response -------|            |                            |
|                              |            |                            |
|-- Group OSCORE Request --+-->|            |                            |
| (kid: 0, gid: abcd0000)  \--------------->|                            |
|                              |            |                            |
|<--- Group OSCORE Response ---|            |                            |
|           (kid: 1)           |            |                            |
|                              |            |                            |
|<--- Group OSCORE Response ----------------|                            |
|           (kid: 2)           |            |                            |
|              ...             |            |                            |
```

Figure 1: Protocol Overview.

2.1. Pre-Conditions

Using Group OSCORE requires both the Client and the Resource Servers to have previously joined an 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 may 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.

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. Access Token Retrieval

This profile requires that the Client retrieves an Access Token from the AS for the resource(s) it wants to access on each of the RSs, using the /token endpoint, as specified in Section 5.6 of [I-D.ietf-ace-oauth-authz]. In a general case, it can be assumed that different RSs are associated to different ASs, even if the RSs are members of a same OSCORE group.

In the Access Token request to the AS, the Client MUST include the Group Identifier of the OSCORE group and its own Sender ID in that group. The AS MUST include these pieces of information in the Access Token and in the Access Token response to the Client.

Furthermore, in the Access Token request to the AS, the Client MUST also include: its own public key, associated to the private signing key used in the OSCORE group; and a signature computed with such private key, over a quantity uniquely related to the secure communication association between the Client and the AS. Finally, the AS MUST include the public key indicated by the client in the Access Token.

To gain knowledge of the AS in charge of a resource hosted at a RS, the Client MAY first send an initial Unauthorized Resource Request message to that RS. Then, the RS denies the request and replies to the Client by specifying the address of its AS, as defined in Section 5.1 of [I-D.ietf-ace-oauth-authz]. The Access Token request and response MUST be confidentiality-protected and ensure authenticity. This profile RECOMMENDS the use of OSCORE between the Client and the AS, but TLS [RFC5246][RFC8446] or DTLS [RFC6347][I-D.ietf-tls-dtls13] MAY be used additionally or instead.

2.3. Access Token Posting

After having retrieved the Access Token from the AS, the Client generates a nonce N1 and posts both the Access Token and N1 to the RS, using the /authz-info endpoint and mechanisms specified in Section 5.8 of [I-D.ietf-ace-oauth-authz] and Content-Format = application/ace+cbor.

If the Access Token is valid, the RS replies to this POST request with a 2.01 (Created) response with Content-Format = application/ace+cbor, which contains a nonce N2 in a CBOR map. Also, the RS sets the ID Context in the pairwise OSCORE Security Context (see Section 3 of [RFC8613]) to the Group Identifier of the OSCORE group specified in the Access Token, concatenated with N1 concatenated with N2.

Then, the RS derives the complete pairwise OSCORE Security Context associated with the received Access Token, following Section 3.2 of [RFC8613]. During the derivation process, the RS uses the ID Context above, the nonces N1 and N2, and the parameters in the Access Token. The derivation process uses also the Master Secret of the OSCORE group, that the RS knows as a group member, as well as the Sender ID of the Client in the OSCORE group, which is specified in the Access Token. This ensures that the pairwise OSCORE Security Context is securely bound to the Group OSCORE Security Context of the OSCORE group.

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 public key of the Client in that group.

After having received the nonce N2, the Client sets the ID Context in its pairwise OSCORE Security Context (see Section 3 of [RFC8613]) to the Group Identifier of the OSCORE group concatenated with N1 concatenated with N2. Then, the Client derives the complete pairwise OSCORE Security Context, following Section 3.2 of [RFC8613]. During the derivation process, the Client uses the ID Context above, the nonces N1 and N2, plus the parameters received from the AS. The derivation process uses also the Master Secret of the OSCORE group, that the Client knows as a group member, as well as its own Sender ID in the OSCORE group.

When the Client communicates with the RS using the pairwise OSCORE Security Context, the RS achieves proof-of-possession of the credentials bound to the Access Token. Also, the RS verifies that the Client is a legitimate member 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 the exact group member with the same Sender ID associated to the Access Token. This occurs when verifying a request protected with Group OSCORE, since it embeds a countersignature computed also over the Client's Sender ID included in the message.

2.4. Secure Communication

The Client can send a request protected with OSCORE to the RS. This message may contain the ID Context value, i.e. the Group Identifier of the OSCORE group concatenated with N1 concatenated with N2. If the request is correctly verified, then the RS stores the pairwise OSCORE Security Context, and uses it to protect the possible response, as well as further communications with the Client, until the Access Token expires. This pairwise OSCORE Security Context is discarded if the same Access Token is re-used to successfully derive a new pairwise OSCORE Security Context. Once the Client has received a valid secure response, it does not continue to include the ID Context value in following requests.

As discussed in Section 2 of [I-D.ietf-ace-oscore-profile], the use of random nonces N1 and N2 during the exchange between the Client and the RS prevents the reuse of AEAD nonces and keys with different messages, in case of re-derivation of the pairwise OSCORE Security Context both for Clients and Resource Servers from an old non-expired Access Token, e.g. in case of reboot of either the Client or the RS.

Furthermore, the Client can send a request protected with Group OSCORE [I-D.ietf-core-oscore-groupcomm]. This can be a unicast request addressed to the RS, or a multicast request addressed to 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]. The RS may send a response back to the Client, protecting it by means of the same Group OSCORE Security Context.

3. Client-AS Communication

This section details the Access Token POST Request that the Client sends to the /token endpoint of the AS, as well as the related Access Token response.

Section 3.2 of [RFC8613] defines how to derive a pairwise OSCORE Security Context based on a shared Master Secret and a set of other parameters, established between the OSCORE client and server.

The Client receives these pieces of information from the AS during the exchange described in this section. In particular, the proof-of-possession key (pop-key) provisioned by the AS MUST be used to build the Master Secret in OSCORE (see Section 4.3 and Section 4.4).

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

The Client-to-AS request is specified in Section 5.6.1 of [I-D.ietf-ace-oauth-authz]. 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 [I-D.ietf-ace-oscore-profile]), with the following additional parameters that MUST be included in the payload.

o 'context_id', defined in Section 3.1.1 of this specification. This parameter includes the Group ID of the OSCORE group that the Client has previously joined and wants to use to communicate with the RS.

o 'salt', defined in Section 3.1.2 of this specification. This parameter includes the Sender ID that the Client has received in the OSCORE group, whose identifier is indicated in the 'context_id' parameter above, upon previously joining it. That is, its value is the Sender ID that the Client uses to communicate in the OSCORE group, whereas it does not relate to the Sender ID to be assigned for use in the pairwise OSCORE Security Context with the RS.

o 'client_cred', defined in Section 3.1.3 of this specification. This parameter includes the public key associated to the signing private key that the Client uses in the OSCORE group, whose identifier is indicated in the 'context_id' parameter above.

o 'client_cred_verify', defined in Section 3.1.4 of this specification. This parameter includes a signature computed by the Client, by using the private key associated to the public key in the 'client_cred' parameter above. This allows the AS to verify that the Client indeed owns the private key associated to the public key in 'client_cred', as its alleged identity credential within the OSCORE group. The information to be signed MUST be the byte representation of a quantity that uniquely represents the secure communication association between the Client and the AS. It is RECOMMENDED that the Client considers the following as information to sign.

* If the Client and the AS communicate over (D)TLS, the information to sign is an exporter value computed as defined in Section 7.5 of [RFC8446], for which a common exporter label has to be agreed between the Client and the AS.

* If the Client and the AS communicate over OSCORE, the information to sign is the output PRK of a HKDF-Extract step [RFC5869], i.e. PRK = HMAC-Hash(salt, IKM). In particular, 'salt' takes (x1 | x2), where x1 is the ID Context of the OSCORE Security Context between the Client and the AS, x2 is the Sender ID of the Client in that Context, and | denotes byte string concatenation. Also, 'IKM' is the OSCORE Master Secret of the OSCORE Security Context between the Client and the AS. The HKDF MUST be one of the HMAC-based HKDF [RFC5869] algorithms defined for COSE [RFC8152]. HKDF SHA-256 is mandatory to implement.

An example of such a request, in CBOR diagnostic notation without the tag and value abbreviations is reported in Figure 2.

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
{
  "audience" : "tempSensor4711",
  "scope" : "read",
  "salt" : h'00',
  "context_id" : h'abcd0000',
  "client_cred" : {
    "COSE_Key" : {
      "kty" : EC2,
      "crv" : P-256,
      "x" : h'd7cc072de2205bdc1537a543d53c60a6acb62eccd890c7fa
              27c9e354089bbe13',
      "y" : h'f95e1d4b851a2cc80fff87d8e23f22afb725d535e515d020
              731e79a3b4e47120'
    }
  },
  "client_cred_verify" : h'...'
}
```

Figure 2: Example C-to-AS POST /token request for an Access Token bound to a symmetric key.

Later on, the Client may want to update its current access rights, without changing the existing pairwise OSCORE Security Context with

the RS. In this case, like in the OSCORE profile of ACE (see Section 3.1 of [I-D.ietf-ace-oscore-profile]), the Client MUST include in its POST request to the /token endpoint a req_cnf object, where the 'kid' field carries the Client's identifier, that was assigned by the AS as per Section 3.2. That is, the Client's identifier is the value of the 'clientId' parameter in the OSCORE Security Context object of the 'cnf' parameter, in the AS-to-C Access Token response providing the original Access Token (see Section 3.2).

The AS can use this identifier to determine the shared secret for preparing the proof-of-possession Access Token. Therefore, the received value MUST identify a symmetric key that was previously generated by the AS, as a shared secret for communications between the Client and the RS. In particular, the AS MUST verify that the received value identifies a proof-of-possession key and Access Token that have previously been issued to the requesting Client. If that is not the case, the Client-to-AS request MUST be declined with the error code 'invalid_request', as defined in Section 5.6.3 of [I-D.ietf-ace-oauth-authz].

This POST request for updating the rights of an Access Token MUST NOT include the parameters 'salt', 'context_id', 'client_cred' and 'client_cred_verify'.

An example of such a request, in CBOR diagnostic notation without the tag and value abbreviations is reported in Figure 3.

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
{
  "audience" : "tempSensor4711",
  "scope" : "read",
  "req_cnf" : {
    "kid" : 'myclient'
  }
}
```

Figure 3: Example C-to-AS POST /token request for updating rights to an Access Token bound to a symmetric key.

3.1.1. 'context_id' Parameter

The 'context_id' parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.6.1. of [I-D.ietf-ace-oauth-authz]. 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.

3.1.2. 'salt' Parameter

The 'salt' parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.6.1. of [I-D.ietf-ace-oauth-authz]. This parameter provides a value that the Client wishes to use as salt with the RS, for deriving cryptographic key material. Its exact content is profile specific.

3.1.3. 'client_cred' Parameter

The 'client_cred' parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.6.1. of [I-D.ietf-ace-oauth-authz]. This parameter provides an asymmetric key that the Client wishes to use as its own public key, but which is not used as proof-of-possession key.

This parameter follows the syntax of the 'cnf' claim from Section 3.1 of [I-D.ietf-ace-cwt-proof-of-possession] when including Value Type "COSE_Key" (1) and specifying an asymmetric key. Alternative Value Types defined in future specifications are fine to consider if indicating a non-encrypted asymmetric key.

3.1.4. 'client_cred_verify' Parameter

The 'client_cred_verify' parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.6.1. of [I-D.ietf-ace-oauth-authz]. This parameter provides a signature computed by the Client to prove the possession of its own private key.

3.2. AS-to-C: Access Token

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 MUST verify the signature in the 'client_cred_verify' parameter, by using the public key specified in the 'client_cred' parameter. If the verification fails, the AS considers the Client request invalid. The AS does not perform this operation when asked to update a previously released Access Token.

If all verifications are successful, the AS responds as defined in Section 5.6.2 of [I-D.ietf-ace-oauth-authz]. If the Client request was invalid, or not authorized, the AS returns an error response as described in Section 5.6.3 of [I-D.ietf-ace-oauth-authz].

The AS can signal that the use of OSCORE and Group OSCORE is REQUIRED for a specific Access Token by including the 'profile' parameter with the value "coap_group_oscore" in the Access Token response. This means that the Client MUST use OSCORE and/or Group OSCORE towards all the Resource Servers for which this Access Token is valid.

In particular, the Client MUST follow Section 4.3 to derive the pairwise OSCORE Security Context to use for communications with the RS. Additionally, the Client has already established the related Group OSCORE Security Context to communicate with members of the OSCORE group, upon previously joining that group.

Usually, it is assumed that constrained devices will be pre-configured with the necessary profile, so that this kind of profile negotiation can be omitted.

The Access Token response to the Client is analogous to the one in the OSCORE profile of ACE, as described in Section 3.2 of [I-D.ietf-ace-oscore-profile]. In particular, the AS provides also the following additional information in the OSCORE_Security_Context object, which is defined in Section 3.2.1 of [I-D.ietf-ace-oscore-profile] and included in the 'cnf' parameter (see Section 3.2 of [I-D.ietf-ace-oauth-params]) of the Access Token response.

- o The 'salt' field in the OSCORE_Security_Context object MUST be present. The field MUST contain the value of the 'salt' parameter from the Access Token request received from the Client.

- o The 'contextId' field in the OSCORE_Security_Context object MUST be present. The field MUST contain the value of the 'context_id' parameter from the Access Token request received from the Client.

The same parameters MUST be included as metadata of the issued Access Token. This profile RECOMMENDS the use of CBOR web tokens (CWT) as specified in [RFC8392]. If the Access Token is a CWT, the same OSCORE_Security_Context structure considered above MUST be placed in the 'cnf' claim of the Access Token.

Furthermore, the AS MUST include also the public key of the client specified in the 'client_cred' parameter of the Token Request as metadata of the issued Access Token. If the Access Token is a CWT, the content of the 'client_cred' parameter MUST be placed in the 'client_cred' claim of the Access Token, defined in Section 3.2.1 of this specification.

As discussed in Section 3.2 of [I-D.ietf-ace-oscore-profile], collisions of client identifiers may appear in the RS, in case a

resource is associated to multiple ASs. In such a case, the RS needs to have a mechanism in place to disambiguate identifiers or mitigate the effect of their collision.

Figure 4 shows an example of such an AS response, in CBOR diagnostic notation without the tag and value abbreviations.

```
     Header: Created (Code=2.01)
     Content-Type: "application/ace+cbor"
     Payload:
     {
       "access_token" : h'a5037674656d7053656e73 ...'
        (remainder of access token omitted for brevity),
       "profile" : "coap_group_oscore",
       "expires_in" : 3600,
       "cnf" : {
         "OSCORE_Security_Context" : {
           "alg" : "AES-CCM-16-64-128",
           "clientId" : b64'qA',
           "serverId" : b64'Qg',
           "ms" : h'f9af838368e353e78888e1426bd94e6f',
           "salt" : h'00',
           "context_id" : h'abcd0000'
         }
       }
     }
```

Figure 4: Example AS-to-C Access Token response with the Group OSCORE profile.

Figure 5 shows an example CWT, containing the necessary OSCORE parameters in the 'cnf' claim, in CBOR diagnostic notation without tag and value abbreviations.

```
        {
          "aud" : "tempSensorInLivingRoom",
          "iat" : "1360189224",
          "exp" : "1360289224",
          "scope" :  "temperature_g firmware_p",
          "cnf" : {
            "OSCORE_Security_Context" : {
              "alg" : "AES-CCM-16-64-128",
              "clientId" : 'client',
              "serverId" : 'server',
              "ms" : h'f9af838368e353e78888e1426bd94e6f',
              "salt" : h'00',
              "context_id" : h'abcd0000'
          },
          "client_cred" : {
            "COSE_Key" : {
              "kty" : EC2,
              "crv" : P-256,
              "x" : h'd7cc072de2205bdc1537a543d53c60a6acb62eccd890c7fa
                      27c9e354089bbe13',
              "y" : h'f95e1d4b851a2cc80fff87d8e23f22afb725d535e515d020
                      731e79a3b4e47120'
            }
          }
        }
```

Figure 5: Example CWT with OSCORE parameters (CBOR diagnostic notation).

The same CWT as in Figure 5 and encoded in CBOR is shown in Figure 6, using the value abbreviations defined in [I-D.ietf-ace-oauth-authz] and [I-D.ietf-ace-cwt-proof-of-possession], and with 12 as value abbreviation for the 'client_cred' claim.

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

```
A5                                     # map(5)
   03                                  # unsigned(3)
   76                                  # text(22)
      74656D7053656E736F72496E4C6976696E67526F6F6D
                                       # "tempSensorInLivingRoom"
   06                                  # unsigned(6)
   1A 5112D728                         # unsigned(1360189224)
   04                                  # unsigned(4)
   1A 51145DC8                         # unsigned(1360289224)
   09                                  # unsigned(9)
```

```
      78 18                                 # text(24)
         74656D70657261747572655F67206669726D776172655F70
                                            # "temperature_g firmware_p"
      08                                    # unsigned(8)
      A1                                    # map(1)
         04                                 # unsigned(4)
         A6                                 # map(6)
            05                              # unsigned(5)
            0A                              # unsigned(10)
            02                              # unsigned(2)
            46                              # bytes(6)
               636C69656E74                 # "client"
            03                              # unsigned(3)
            46                              # bytes(6)
               736572766572                 # "server"
            01                              # unsigned(1)
            50                              # bytes(16)
               F9AF838368E353E78888E1426BD94E6F
            06                              # unsigned(6)
            41                              # bytes(1)
               00
            07                              # unsigned(7)
            44                              # bytes(4)
               ABCD0000
      0C                                    # unsigned(12)
      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)
               D7CC072DE2205BDC1537A543D53C60A6ACB62ECCD890C7FA27C9
               E354089BBE13
            22                              # negative(2)
            58 20                           # bytes(32)
               F95E1D4B851A2CC80FFF87D8E23F22AFB725D535E515D020731E
               79A3B4E47120
```

Figure 6: Example CWT with OSCORE parameters.

If the Client has requested an update to its access rights with reference to the same pairwise OSCORE Security Context, which is valid and authorized, the AS MUST omit the 'cnf' parameter in the Access Token response, MUST omit the 'client_cred' claim in the Access Token, and MUST include the Client identifier in the 'kid'

field of the 'cnf' claim of the Access Token. The Client identifier needs to be provisioned, in order for the RS to identify the previously generated pairwise OSCORE Security Context.

Figure 7 shows an example of such an AS response, in CBOR diagnostic notation without the tag and value abbreviations.

```
     Header: Created (Code=2.01)
     Content-Type: "application/ace+cbor"
     Payload:
     {
       "access_token" : h'a5037674656d7053656e73 ...'
        (remainder of access token omitted for brevity),
       "profile" : "coap_group_oscore",
       "expires_in" : 3600
     }
```

Figure 7: Example AS-to-C Access Token response with the Group OSCORE profile, for update of access rights.

Figure 8 shows an example CWT, containing the necessary OSCORE parameters in the 'cnf' claim for update of access rights, in CBOR diagnostic notation without tag and value abbreviations.

```
     {
       "aud" : "tempSensorInLivingRoom",
       "iat" : "1360189224",
       "exp" : "1360289224",
       "scope" :  "temperature_h",
       "cnf" : {
         "kid" : b64'qA'
       }
     }
```

Figure 8: Example CWT with OSCORE parameters for update of access rights.

3.2.1. **Client Credential Claim**

The 'client_cred' claim provides an asymmetric key that the Client owning the Access Token wishes to use as its own public key, but which is not used as proof-of-possession key.

This parameter follows the syntax of the 'cnf' claim from Section 3.1 of [I-D.ietf-ace-cwt-proof-of-possession] when including Value Type "COSE_Key" (1) and specifying an asymmetric key. Alternative Value Types defined in future specifications are fine to consider if indicating a non-encrypted asymmetric key.

4. Client-RS Communication

This section details the POST request and response to the /authz-info endpoint between the Client and the RS. With respect to the exchanged messages and their content, the Client and the RS perform as defined in Section 4 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

That is, the Client generates a nonce N1 and posts it to the RS, together with the Access Token that includes the material provisioned by the AS. Then, the RS generates a nonce N2, and derives a pairwise OSCORE Security Context as described Section 3.2 of [RFC8613]. In particular, it uses the two nonces established with the Client and two shared secrets, together with additional pieces of information specified in the Access Token.

The proof-of-possession required to bind the Access Token to the Client is implicitly performed by generating the pairwise OSCORE Security Context using the pop-key as part of the OSCORE Master Secret, for both the Client and the RS. In addition, the derivation of the pairwise OSCORE Security Context takes as input also information related to the OSCORE group, i.e. the Master Secret and Group Identifier of the group, as well as the Sender ID of the Client in the group. Hence, the derived pairwise OSCORE Security Context is also securely bound to the Group OSCORE Security Context of the OSCORE Group.

Therefore, an attacker using a stolen Access Token cannot generate a valid pairwise OSCORE Security Context and thus cannot prove possession of the pop-key. Also, if a Client legitimately owns an Access Token but has not joined the OSCORE group, that Client cannot generate a valid pairwise OSCORE Security Context either, since it lacks the Master Secret used in the OSCORE group.

Finally, a Client C1 is supposed to obtain a valid Access Token from the AS, as including the public key associated to its own signing key used in the OSCORE group, together with its own Sender ID in that OSCORE group (see Section 3.1). This makes it possible for the RS receiving an Access Token to verify with the Group Manager of that OSCORE group whether such a Client has indeed that Sender ID and that public key 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) public key; ii) successfully posting the Access Token to RS; and then iii) using only the pairwise OSCORE Security Context to

communicate with RS to legitimately perform authorized actions, while blaming C1 for the consequences.

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

The Client MUST generate a nonce N1 and post it to the /authz-info endpoint of the RS together with the Access Token, as defined in Section 4.1 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

The same recommendations, considerations and behaviors defined in Section 4.1 of [I-D.ietf-ace-oscore-profile] hold for this specification.

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

The RS MUST verify the validity of the Access Token as defined in Section 4.2 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile], with the following additions.

- The RS checks that the OSCORE_Security_Context object in the 'cnf' claim of the Access Token includes the 'salt' parameter.

- The RS checks that the OSCORE_Security_Context object in the 'cnf' claim of the Access Token includes the 'context_id' parameter. Also, the RS checks that the value of the 'context_id' parameter coincides with the one of the group identifier of the OSCORE group associated to the resources targeted by the scope in the Access Token.

- The RS checks that the 'client_cred' claim is included in the Access Token, unless this is intended to update and supersede an active Access Token for that same Client. The RS considers the content of the 'client_cred' claim (if present) as the public key associated to the signing private key that the Client uses in the OSCORE group, which is identified by the 'context_id' parameter above. The RS MAY additionally request the Group Manager of the OSCORE group for the public key of that Client, as described in [I-D.ietf-ace-key-groupcomm-oscore]. In such a case, the RS MUST check that the key retrieved from the Group Manager matches the one retrieved from the 'client_cred' claim.

If any of the checks above fails, the RS MUST consider the Access Token non valid, and MUST respond to the Client with an error response code equivalent to the CoAP code 4.00 (Bad Request).

If the Access Token is valid and further checks on its content are successful, the RS MUST generate a nonce N2 and include it in the

2.01 (Created) response to the Client, as defined in Section 4.2 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

Further recommendations, considerations and behaviors defined in Section 4.2 of [I-D.ietf-ace-oscore-profile] hold for this specification.

4.3. OSCORE Setup - Client Side

Once having received the 2.01 (Created) response from the RS, following the POST request to the authz-info endpoint, the Client MUST extract the nonce N2 from the 'cnonce' parameter and the client identifier from the 'clientId' parameter (if present) in the CBOR map in the payload of the response.

Note that, if present in the 2.01 (Created) response, the 'clientId' parameter supersedes the analogous parameter possibly provided by the AS to C in Section 3.2. Also, note that this identifier is used by C as Sender ID in the pairwise OSCORE Security Context to be established with the RS, and is different as well as unrelated to the Sender ID of C in the OSCORE group.

Then, the Client performs the following actions, in order to set up and fully derive the pairwise OSCORE Security Context for communicating with the RS.

- o The Client MUST set the ID Context of the pairwise OSCORE Security Context as the concatenation of: the Group Identifier GID of the OSCORE group; the nonce N1; and the nonce N2. The concatenation occurs in this order: ID Context = GID | N1 | N2, where | denotes byte string concatenation.

- o The Client MUST set the Master Salt of the pairwise OSCORE Security Context as the concatenation of MSalt, N1, N2 and GMSalt, where: i) MSalt is the Sender ID that the Client has in the OSCORE group; while ii) GMSalt is the (optional) Master Salt in the Group OSCORE Security Context, which is known to the Client as a member of the OSCORE group. The concatenation occurs in this order: Master Salt = MSalt | N1 | N2 | GMSalt, where | denotes byte string concatenation.

- o The Client MUST set the Master Secret of the pairwise OSCORE Security Context to the concatenation of MSec and GMSec, where: i) MSec is the value of the 'ms' parameter in the OSCORE_Security_Context object of the 'cnf' parameter, received from the AS in Section 3.2; while ii) GMSec is the Master Secret of the Group OSCORE Security Context, which is known to the Client as a member of the OSCORE group.

- The Client MUST set the Recipient ID as indicated in the corresponding parameter received from the AS in Section 3.2, i.e. to the value of the 'serverId' parameter in the OSCORE_Security_Context object of the 'cnf' parameter.

- The Client MUST set the AEAD Algorithm, HKDF, and Replay Window as indicated in the corresponding parameters received from the AS in Section 3.2, if present in the OSCORE_Security_Context object of the 'cnf' parameter. In case these parameters are omitted, the default values are used as described in Section 3.2 of [RFC8613].

- The client MUST set the Sender ID as indicated in the 'clientId' parameter from the 2.01 (Created) response, if present. Otherwise, the Client MUST set the Sender ID as indicated in the response from the AS in Section 3.2, i.e. to the value of the 'clientId' parameter in the OSCORE_Security_Context object of the 'cnf' parameter.

Finally, the client MUST derive the complete pairwise OSCORE Security Context following Section 3.2.1 of [RFC8613].

From then on, when communicating with the RS to access the resources as specified by the authorization information, the Client MUST use the newly established pairwise OSCORE Security Context or the Group OSCORE Security Context of the OSCORE Group where both the Client and the RS are members.

If any of the expected parameters is missing (e.g. any of the mandatory parameters from the AS, or 'clientId' both from the AS and in the 2.01 (Created) response from the RS), the Client MUST stop the exchange, and MUST NOT derive the pairwise OSCORE Security Context. The Client MAY restart the exchange, to get the correct security material.

Then, the Client uses this pairwise OSCORE Security Context to send requests to RS protected with OSCORE. In the first request sent to the RS, the Client MAY include the kid context if the application needs to, with value the ID Context, i.e. GID concatenated with N1 concatenated with N2. Besides, the Client uses the Group OSCORE Security Context for protecting unicast requests to the RS, or multicast requests to the OSCORE group including also the RS.

4.4. OSCORE Setup - Resource Server Side

After validation of the Access Token as defined in Section 4.2 and after sending the 2.01 (Created) response, the RS performs the following actions, in order to set up and fully derive the pairwise OSCORE Security Context created to communicate with the Client.

- The RS MUST set the ID Context of the pairwise OSCORE Security Context as the concatenation of: the Group Identifier GID of the OSCORE group; the nonce N1; and the nonce N2. The concatenation occurs in this order: ID Context = GID | N1 | N2, where | denotes byte string concatenation.

- The RS MUST set the Master Salt of the pairwise OSCORE Security Context as the concatenation of MSalt, N1, N2 and GMSalt, where: i) MSalt is the Sender ID that the Client has in the OSCORE group, as specified in the 'salt' parameter in the OSCORE_Security_Context object of the 'cnf' claim, included in the Access Token received from the Client (see Section 4.1); while ii) GMSalt is the (optional) Master Salt in the Group OSCORE Security Context, which is known to the RS as a member of the OSCORE group. The concatenation occurs in this order: Master Salt = MSalt | N1 | N2 | GMSalt, where | denotes byte string concatenation.

- The RS MUST set the Master Secret of the pairwise OSCORE Security Context to the concatenation of MSec and GMSec, where: i) MSec is the value of the 'ms' parameter in the OSCORE_Security_Context object of the 'cnf' claim, included in the Access Token received from the Client (see Section 4.1); while ii) GMSec is the Master Secret of the Group OSCORE Security Context, which is known to the RS as a member of the OSCORE group.

- The RS MUST set the Sender ID of the pairwise OSCORE Security Context from the corresponding parameter received from the Client in the Access Token (see Section 4.1), i.e. to the value of the 'serverId' parameter in the OSCORE_Security_Context object of the 'cnf' claim.

- The RS MUST set the Recipient ID of the pairwise OSCORE Security Context from either what it indicated in the 2.01 (Created) response if included (see Section 4.2 of [I-D.ietf-ace-oscore-profile]), or from the corresponding parameter received from the Client in the Access Token (see Section 4.1), i.e. to the value of the 'clientId' parameter in the OSCORE_Security_Context object of the 'cnf' claim.

- The RS MUST set the AEAD Algorithm, HKDF, and Replay Window from the corresponding parameters received from the Client in the Access Token (see Section 4.1), if present in the OSCORE_Security_Context object of the 'cnf' claim. In case these parameters are omitted, the default values are used as described in Section 3.2 of [RFC8613].

Finally, the RS MUST derive the complete pairwise OSCORE Security Context following Section 3.2.1 of [RFC8613].

Once having completed the derivation above, the RS MUST associate the authorization information from the Access Token with the just established pairwise OSCORE Security Context.

Furthermore, the RS MUST associate the authorization information from the Access Token with the tuple (GID, MSalt, PKey), where GID is the Group Identifier of the OSCORE Group, while MSalt and PKey are the Sender ID and the public key that the Client has in that OSCORE group, respectively. The RS MUST keep this association up-to-date over time.

Then, the RS uses this pairwise OSCORE Security Context to verify requests from and send responses to the Client protected with OSCORE, when this Security Context is used. If OSCORE verification fails, error responses are used, as specified in Section 8 of [RFC8613]. Besides, the RS uses the Group OSCORE Security Context to verify (multicast) requests from and send responses to the Client protected with Group OSCORE. If Group OSCORE verification fails, error responses are used, as specified in Section 6 of [I-D.ietf-core-oscore-groupcomm]. Additionally, for every incoming request, if OSCORE or Group OSCORE verification succeeds, the verification of access rights is performed as described in Section 4.5.

After the expiration of the Access Token related to a pairwise OSCORE Security Context and to a Group OSCORE Security Context, the RS MUST NOT use the pairwise OSCORE Security Context and MUST respond with an unprotected 4.01 (Unauthorized) error message. Also, if the Client uses the Group OSCORE Security Context to send a request for any resource intended for OSCORE group members and that requires an active Access Token, the RS MUST respond with a 4.01 (Unauthorized) error message protected with the Group OSCORE Security Context.

4.5. Access Rights Verification

The RS MUST follow the procedures defined in Section 5.8.2 of [I-D.ietf-ace-oauth-authz]. If an RS receives an OSCORE-protected request from a Client, the RS processes it according to [RFC8613]. If an RS receives a Group OSCORE-protected request from a Client, the RS processes it according to [I-D.ietf-core-oscore-groupcomm].

If the OSCORE or Group OSCORE verification succeeds, and the target resource requires authorization, the RS retrieves the authorization information from the Access Token associated to the pairwise OSCORE Security Context and to the Group OSCORE Security Context. Then, the RS MUST verify that the authorization information covers the resource and the action requested by the Client.

The response code MUST be 4.01 (Unauthorized) in case the Client has not used either of the two Security Contexts associated with the Access Token, or if the RS has no valid Access Token for the Client. If the RS has an Access Token for the Client but not for the resource that was requested, the RS MUST reject the request with a 4.03 (Forbidden). If the RS has an Access Token for the Client but it does not cover the action that was requested on the resource, the RS MUST reject the request with a 4.05 (Method Not Allowed).

5. Secure Communication with the AS

As specified in the ACE framework (Section 5.7 of [I-D.ietf-ace-oauth-authz]), the requesting entity (RS and/or Client) and the AS communicate via the /introspection or /token endpoint. The use of CoAP and OSCORE for this communication is RECOMMENDED in this profile. Other protocols (such as HTTP and DTLS or TLS) MAY be used instead.

If OSCORE is used, the requesting entity and the AS are expected to have pre-established security contexts in place. How these security contexts are established is out of the scope of this profile. Furthermore the requesting entity and the AS communicate using OSCORE ([RFC8613]) through the /introspection endpoint as specified in Section 5.7 of [I-D.ietf-ace-oauth-authz], and through the /token endpoint as specified in Section 5.6 of [I-D.ietf-ace-oauth-authz].

6. Discarding the Security Context

The Client and the RS MUST follow what is defined in Section 6 of [I-D.ietf-ace-oscore-profile] about discarding the pairwise OSCORE 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 2.1 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 and post it to the RS, in order to establish a new pairwise OSCORE Security Context and bind it to the Group OSCORE Security Context obtained upon re-joining the group.

7. CBOR Mappings

The new parameters and claims defined in this document MUST be mapped to CBOR types as specified in Figure 9, using the given integer abbreviation for the map key.

Parameter name	CBOR Key	Value Type
context_id	TBD1	bstr
salt	TBD2	bstr
client_cred	TBD3	map
client_cred_verify	TBD4	bstr

Figure 9: CBOR mappings for new parameters.

8. Security Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [I-D.ietf-ace-oauth-authz]. Thus the general security considerations from the ACE framework also apply to this profile.

This specification inherits the security considerations from the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile]. Also, the general security considerations about OSCORE [RFC8613] hold for this document, as to the specific use of OSCORE according to this profile.

Furthermore, the general security considerations about Group OSCORE [I-D.ietf-core-oscore-groupcomm] hold for this document, 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, for group requests and responses. To ensure source authentication of messages, Group OSCORE uses digital countersignatures that group members embed in their own transmitted messages.

9. Privacy Considerations

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework [I-D.ietf-ace-oauth-authz]. Thus the general privacy considerations from the ACE framework also apply to this profile.

As this profile uses OSCORE and Group OSCORE, the privacy considerations from [RFC8613] and [I-D.ietf-core-oscore-groupcomm] apply to this document as well.

This profile also inherits the privacy considerations from the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

10. IANA Considerations

This document has the following actions for IANA.

10.1. ACE Profile Registry

IANA is asked to enter the following value into the "ACE Profile" Registry defined in Section 8.7 of [I-D.ietf-ace-oauth-authz].

- Profile name: coap_group_oscore

- Profile Description: Profile to secure communications between constrained nodes using the Authentication and Authorization for Constrained Environments framework by: i) establishing a Pairwise OSCORE Security Context and enabling OSCORE communication between two members of an OSCORE group; ii) 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.

- Profile ID: TBD (value between 1 and 255)

- Change Controller: IESG

- Specification Document(s): [[this document]]

10.2. OAuth Parameters Registry

IANA is asked to enter the following values into the "OAuth Parameters" Registry defined in Section 11.2 of [RFC6749].

- Name: "context_id"

- Parameter Usage Location: token request

- Change Controller: IESG

- Reference: Section 3.1.1 of [[this document]]

- Name: "salt"

- Parameter Usage Location: token request

- Change Controller: IESG

- Reference: Section 3.1.2 of [[this document]]

- Name: "client_cred"

- Parameter Usage Location: token request

- Change Controller: IESG

- Reference: Section 3.1.3 of [[this document]]

- Name: "client_cred_verify"

- Parameter Usage Location: token request

- Change Controller: IESG

- Reference: Section 3.1.4 of [[this document]]

10.3. OAuth Parameters CBOR Mappings Registry

IANA is asked to enter the following values into the "OAuth Parameters CBOR Mappings" Registry defined in Section 8.9 of [I-D.ietf-ace-oauth-authz].

- Name: "context_id"

- CBOR Key: TBD1

- Change Controller: IESG

- Reference: Section 3.1.1 of [[this document]]

- Name: "salt"

- CBOR Key: TBD2

- Change Controller: IESG

- Reference: Section 3.1.2 of [[this document]]

- Name: "client_cred"

- CBOR Key: TBD3

- Change Controller: IESG

- Reference: Section 3.1.3 of [[this document]]

- Name: "client_cred_verify"

- CBOR Key: TBD4

- Change Controller: IESG

- Reference: Section 3.1.4 of [[this document]]

10.4. CBOR Web Token Claims Registry

IANA is asked to enter the following values into the "CBOR Web Token Claims" Registry defined in Section 9.1 of [RFC8392].

- Claim Name: "client_cred"

- Claim Description: Client Credential

- JWT Claim Name: "N/A"

- Claim Key: TBD5

- Claim Value Type(s): map

- Change Controller: IESG

- Specification Document(s): Section 3.2.1 of [[this document]]

11. References

11.1. Normative References

[I-D.dijk-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", draft-dijk-core-groupcomm-bis-01 (work in progress), July 2019.

[I-D.ietf-ace-cwt-proof-of-possession]
Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", draft-ietf-ace-cwt-proof-of-possession-11 (work in progress), October 2019.

[I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", draft-ietf-ace-key-groupcomm-oscore-02 (work in progress), July 2019.

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-25 (work in progress), October 2019.

[I-D.ietf-ace-oauth-params]
Seitz, L., "Additional OAuth Parameters for Authorization in Constrained Environments (ACE)", draft-ietf-ace-oauth-params-05 (work in progress), March 2019.

[I-D.ietf-ace-oscore-profile]
Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "OSCORE profile of the Authentication and Authorization for Constrained Environments Framework", draft-ietf-ace-oscore-profile-08 (work in progress), July 2019.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", draft-ietf-core-oscore-groupcomm-05 (work in progress), July 2019.

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

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

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

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

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

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

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

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

11.2. Informative References

[I-D.ietf-ace-dtls-authorize]
Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)", draft-ietf-ace-dtls-authorize-08 (work in progress), April 2019.

[I-D.ietf-ace-mqtt-tls-profile]
Sengul, C., Kirby, A., and P. Fremantle, "MQTT-TLS profile of ACE", draft-ietf-ace-mqtt-tls-profile-02 (work in progress), November 2019.

[I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-33 (work in progress), October 2019.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsuess, C., and P. Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", draft-tiloca-core-oscore-discovery-03 (work in progress), July 2019.

[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/info/rfc5246>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7390] Rahman, A., Ed. and E. Dijk, Ed., "Group Communication for the Constrained Application Protocol (CoAP)", RFC 7390, DOI 10.17487/RFC7390, October 2014, <https://www.rfc-editor.org/info/rfc7390>.

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

Appendix A. Profile Requirements

This appendix lists the specifications on this profile based on the requirements of the ACE framework, as requested in Appendix C of [I-D.ietf-ace-oauth-authz].

o (Optional) discovery process of how the Client finds the right AS for an RS it wants to send a request to: Not specified.

o Communication protocol the Client and the RS must use: CoAP.

o Security protocol(s) the Client and RS must use: OSCORE, i.e. establishment of a pairwise OSCORE Security Context and exchange of secure messages; and/or Group OSCORE, i.e. exchange of secure messages by using a pre-established Group OSCORE Security Context.

o How the Client and the RS mutually authenticate: Implicitly by possession of a common OSCORE Security Context (when using OSCORE); and/or explicitly, by possession of a common Group OSCORE Security Context and usage of digital countersignatures (when using Group OSCORE).

o Content-format of the protocol messages: "application/ace+cbor".

o Proof-of-Possession protocol(s) and how to select one; which key types (e.g. symmetric/asymmetric) supported: OSCORE algorithms; pre-established symmetric keys.

o profile identifier: coap_group_oscore

o (Optional) how the RS talks to the AS for introspection: HTTP/CoAP (+ TLS/DTLS/OSCORE).

o How the client talks to the AS for requesting a token: HTTP/CoAP (+ TLS/DTLS/OSCORE).

- o How/if the authz-info endpoint is protected: Security protocol above.

- o (Optional) other methods of token transport than the authz-info endpoint: no.

Acknowledgments

The authors sincerely thank Dave Robin, Jim Schaad and Goeran Selander for their comments and feedback.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC.

Authors' Addresses

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

Email: marco.tiloca@ri.se

Rikard Hoeglund
RISE AB
Isafjordsgatan 22
Kista SE-16440 Stockholm
Sweden

Email: rikard.hoglund@ri.se

Ludwig Seitz
RISE AB
Scheelevagen 17
Lund SE-22370 Lund
Sweden

Email: ludwig.seitz@ri.se

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

Email: francesca.palombini@ericsson.com