

ACE Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: March 25, 2021

F. Palombini  
Ericsson AB  
September 21, 2020

OSCORE Profile of ACE v2  
draft-palombini-ace-oscore-profile-v2-00

## Abstract

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework. It utilizes Object Security for Constrained RESTful Environments (OSCORE) to provide communication security and proof-of-possession for a key owned by the client and bound to an OAuth 2.0 access token. Additionally, this profile provides OSCORE identifiers negotiation between Client and Resource Server.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 25, 2021.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction</a>	<a href="#">2</a>
<a href="#">1.1.</a>	<a href="#">Terminology</a>	<a href="#">3</a>
<a href="#">1.2.</a>	<a href="#">Background</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">Protocol Overview</a>	<a href="#">4</a>
<a href="#">3.</a>	<a href="#">Client-AS Communication</a>	<a href="#">5</a>
<a href="#">3.1.</a>	<a href="#">C-to-AS: POST to token endpoint</a>	<a href="#">5</a>
<a href="#">3.2.</a>	<a href="#">AS-to-C: Access Token</a>	<a href="#">6</a>
<a href="#">4.</a>	<a href="#">Client-RS Communication</a>	<a href="#">10</a>
<a href="#">4.1.</a>	<a href="#">C-to-RS: POST to authz-info endpoint</a>	<a href="#">10</a>
<a href="#">4.2.</a>	<a href="#">RS-to-C: 2.01 (Created)</a>	<a href="#">11</a>
<a href="#">4.3.</a>	<a href="#">OSCORE Setup</a>	<a href="#">12</a>
<a href="#">5.</a>	<a href="#">Security Considerations</a>	<a href="#">12</a>
<a href="#">6.</a>	<a href="#">Privacy Considerations</a>	<a href="#">12</a>
<a href="#">7.</a>	<a href="#">IANA Considerations</a>	<a href="#">13</a>
<a href="#">7.1.</a>	<a href="#">ACE Profile Registry</a>	<a href="#">13</a>
<a href="#">7.2.</a>	<a href="#">OAuth Parameters Registry</a>	<a href="#">13</a>
<a href="#">7.3.</a>	<a href="#">OAuth Parameters CBOR Mappings Registry</a>	<a href="#">13</a>
<a href="#">7.4.</a>	<a href="#">OSCORE Security Context Parameters Registry</a>	<a href="#">14</a>
	<a href="#">Acknowledgments</a>	<a href="#">14</a>
<a href="#">9.</a>	<a href="#">References</a>	<a href="#">14</a>
<a href="#">9.1.</a>	<a href="#">Normative References</a>	<a href="#">14</a>
<a href="#">9.2.</a>	<a href="#">Informative References</a>	<a href="#">15</a>
	<a href="#">Author's Address</a>	<a href="#">15</a>

## [1.](#) Introduction

An OSCORE profile of ACE is defined in [[I-D.ietf-ace-oscore-profile](#)]. That profiles describes how to set up OSCORE between nodes, while the OSCORE Sender and Recipient Identifiers, i.e. the identifiers of the OSCORE keying material, are assigned by the Authorization Server. This has some limitations, especially if the OSCORE profile is used in conjunction with other mechanisms that also derive identifiers, in which case there needs to be a mechanism in place to avoid collisions.

This document describes a new OSCORE profile that builds on [[I-D.ietf-ace-oscore-profile](#)], and adds a mechanism for negotiating identifiers between Client and Resource Server.

### [1.1.](#) Terminology

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

Readers are expected to be familiar with the terms and concepts described in [[I-D.ietf-ace-oauth-authz](#)] [[I-D.ietf-ace-oscore-profile](#)], such as Authorization Server (AS) and Resource Server (RS).

Readers are expected to be familiar with the terms and concepts described in [[RFC8613](#)], especially on the use of Sender, Recipient and Context Identifiers.

### [1.2.](#) Background

TODO: introduce OSCORE Sender and Recipient Identifiers and how they are used in OSCORE.

The OSCORE profile defined in [[I-D.ietf-ace-oscore-profile](#)] specifies that the AS assigns and sends the OSCORE Sender and Recipient Identifiers to both Client and RS, together with the rest of the input material to derive the complete OSCORE Security Context. That is done by including these identifiers in the Access Token and Access Information response to the Client. The access token containing these identifiers is also forwarded to the RS by the Client.

This works with a number of requirements: the OSCORE profile states that if other authentication mechanisms are used to set up OSCORE between the same client and RS, that do not rely on an AS assigning identifiers, collisions may happen and need to be mitigated. Such mitigation mechanisms also need to be used if a different AS (which does not synchronize identifiers with the first AS) or authentication

protocol is used to set up OSCORE between the same RS and other clients. A mitigation example would be to use distinct namespaces of context identifiers for different authentication mechanisms or authentication servers. Another solution would be to use longer random identifiers. A third possible solution, acceptable if collisions are not expected to be numerous, would be to rely on trial and error of security contexts when receiving a message.

These solutions have the drawback of requiring either some sort of agreement between different authentication mechanisms using disjoint namespaces for the identifiers, and/or longer identifiers to be used,

which leads to larger message sizes, or additional processing on the RS.

This document specifies an OSCORE profile that adds a mechanism to assign identifiers on top of the current OSCORE profile, and that allows to set up identifiers without collisions, even when other authentication mechanisms or non-synchronized AS are used. What specified in [[I-D.ietf-ace-oscore-profile](#)] applies to this profile as well, with the modifications reported in the following sections. Although defining a separate profile, the reader is advised that the [[I-D.ietf-ace-oscore-profile](#)] needs to be read side by side with this specification, to understand it.

## [2.](#) Protocol Overview

This section defines the additions to Section 2 of [[I-D.ietf-ace-oscore-profile](#)].

Once the client has retrieved the access token, and generated the nonce N1, the Client also generates a Recipient Identifier ID1 for use with the keying material associated to the RS. The client posts the token, N1 and its Recipient ID 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.

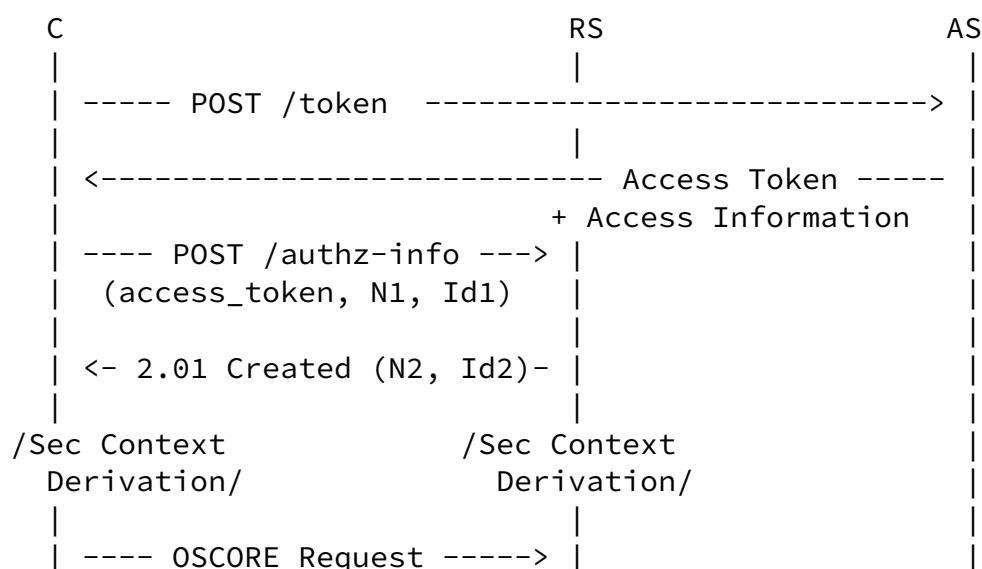
If the access token is valid, the RS replies to this request with a 2.01 (Created) response with Content-Format = application/ace+cbor, which contains a nonce N2 and a newly generated Recipient Identifier ID2 for use with the keying material associated to the Client in a

CBOR map. Moreover, the server concatenates the input salt, N1, and N2 to obtain the Master Salt of the OSCORE Security Context (see [section 3 of \[RFC8613\]](#)). The RS then derives the complete Security Context associated with the received token from the Master Salt, the Recipient ID generated by the Client (set as its OSCORE Sender Id), the Recipient ID generated by itself (set as its OSCORE Recipient Id), plus the parameters received in the access token from the AS, following [section 3.2 of \[RFC8613\]](#).

In a similar way, after receiving the nonce N2, the client concatenates the input salt, N1 and N2 to obtain the Master Salt of the OSCORE Security Context. The client then derives the complete Security Context from the Master Salt, the Recipient ID generated by the RS (set as its OSCORE Sender Id), the Recipient ID generated by itself (set as its OSCORE Recipient Id), plus the parameters received from the AS.

After the whole message exchange has taken place, the client can contact the AS to request an update of its access rights, sending a

similar request to the token endpoint that also includes an identifier so that the AS can find the correct OSCORE security material it has previously shared with the Client. This specific identifier, encoded as a byte string, is set by the AS to be unique in the sets of its OSCORE Security Contexts, and is not used as input material to derive the full OSCORE Security Context.



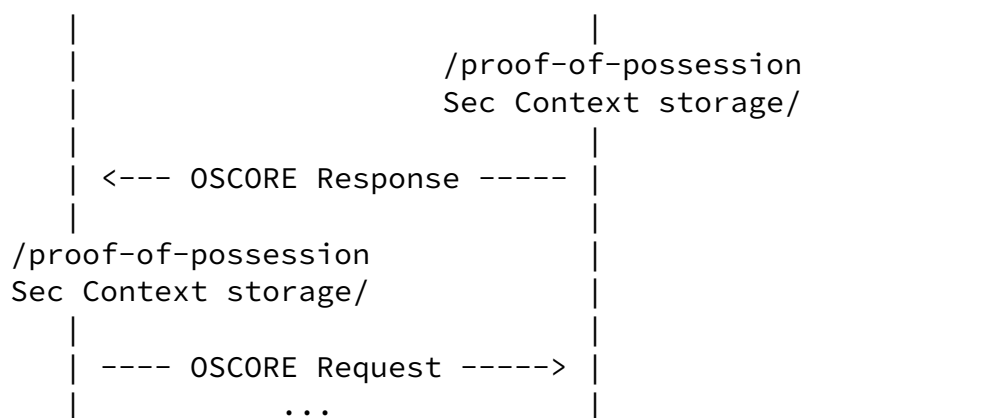


Figure 1: OSCORE Profile v2 Overview

### 3. Client-AS Communication

The following subsections apply modifications to what specified in Section 3 of [[I-D.ietf-ace-oscore-profile](#)].

#### 3.1. C-to-AS: POST to token endpoint

If the client wants to update its access rights without changing an existing OSCORE Security Context, it MUST include in its POST request to the token endpoint a req\_cnf object. The req\_cnf MUST include a kid field carrying a CBOR byte string containing the

OSCORE\_Input\_Material Identifier (assigned as discussed in [Section 3.2](#)).

An example of an update of access rights request, with payload 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:
{
  "req_aud" : "tempSensor4711",
  "scope" : "write",

```

```
"req_cnf" : {  
  "kid" : h'01'  
}
```

Figure 2: Example C-to-AS POST /token request for updating rights to an access token bound to a symmetric key.

### [3.2.](#) AS-to-C: Access Token

The AS can signal that the use of OSCORE is REQUIRED for a specific access token by including the "profile" parameter with the value "coap\_oscore\_2" in the access token response.

The AS MUST send the following data:

- o a master secret
- o an identifier of the OSCORE\_Input\_Material

The AS MUST NOT include `clientId` or `serverId`, in the `OSCORE_Input_Material`, neither in the 'cnf' nor in the claims sets associated with the access token.

The identifier of the `OSCORE_Input_Material` MUST be communicated as the 'id' field in the 'osc' field in the 'cnf' parameter of the access token response (and therefore included in the claims associated with the access token).

We don't assume in this document that a resource is associated to one single AS, since the AS is not tasked with enforcing uniqueness of identifiers for each client requesting a particular resource to a RS. This profile can also be used together with other authentication mechanisms such as EDHOC, see [[I-D.ietf-lake-edhoc](#)].

Figure 3 shows an example of an AS response, with payload in CBOR diagnostic notation without the tag and value abbreviations. The access token has been truncated for readability.

```
Header: Created (Code=2.01)  
Content-Type: "application/ace+cbor"  
Payload:  
{
```

```

"access_token" : h'8343a1010aa2044c53 ...
  (remainder of access token (CWT) omitted for brevity)',
"profile" : "coap_oscure_2",
"expires_in" : "3600",
"cnf" : {
  "osc" : {
    "ms" : h'f9af838368e353e78888e1426bd94e6f',
    "id" : h'01'
  }
}
}

```

Figure 3: Example AS-to-C Access Token response with OSCORE profile 2.

Figure 4 shows an example CWT Claims Set, including the relevant 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" : {
    "osc" : {
      "ms" : h'f9af838368e353e78888e1426bd94e6f',
      "id" : h'01'
    }
  }
}

```

Figure 4: Example CWT Claims Set with OSCORE parameters.

The same CWT Claims Set as in Figure 4, using the value abbreviations defined in [[I-D.ietf-ace-oauth-authz](#)] and [[RFC8747](#)] and encoded in CBOR is shown in Figure 5. 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.



```

63          # text(3)
617564      # "aud"
76          # text(22)
74656D7053656E736F72496E4C6976696E67526F6F6D
          # "tempSensorInLivingRoom"
63          # text(3)
696174      # "iat"
6A          # text(10)
31333630313839323234
          # "1360189224"
63          # text(3)
657870      # "exp"
6A          # text(10)
31333630323839323234
          # "1360289224"
65          # text(5)
73636F7065
          # "scope"
78 18       # text(24)
74656D70657261747572655F67206669726D776172655F70
          # "temperature_g firmware_p"
63          # text(3)
636E66      # "cnf"
A1          # map(1)
63          # text(3)
6F7363      # "osc"
A2          # map(2)
62          # text(2)
6D73        # "ms"
50          # bytes(16)
F9AF838368E353E78888E1426BD94E6F
          # "\xF9\xAF\x83\x83h\xE3S\xE7
          \x88\x88\xE1Bk\xD9No"
62          # text(2)
6964        # "id"
41          # bytes(1)
01          # "\x01"

```

Figure 5: Example CWT Claims Set with OSCORE parameters, CBOR encoded

If the client has requested an update to its access rights using the same OSCORE\_Input\_Material, which is valid and authorized, the AS MUST omit the 'cnf' parameter in the response, and MUST carry the OSCORE\_Input\_Material Identifier in the 'kid' field in the 'cnf' parameter of the token. This identifier needs to be included in the token in order for the RS to identify the correct OSCORE Input Material.

Figure 6 shows an example of such an AS response, with payload in CBOR diagnostic notation without the tag and value abbreviations. The access token has been truncated for readability.

```
Header: Created (Code=2.01)
Content-Type: "application/ace+cbor"
Payload:
{
  "access_token" : h'8343a1010aa2044c53 ...
    (remainder of access token (CWT) omitted for brevity)',
  "profile" : "coap_oscore_2",
  "expires_in" : "3600"
}
```

Figure 6: Example AS-to-C Access Token response with OSCORE profile, for update of access rights.

Figure 7 shows an example CWT Claims Set, 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" : h'01'
  }
}
```

Figure 7

### [3.2.1.](#) OSCORE\_Input\_Material Object

The following parameter is added to the OSCORE\_Input\_Material object defined in Section 3.2.1 of [\[I-D.ietf-ace-oscore-profile\]](#).

name	CBOR label	CBOR type	registry	description
identifier	8	byte string		OSCORE Input Material Identifier

Figure 8: OSCORE\_Input\_Material Additional Parameters

id: This parameter identifies the OSCORE\_Input\_Material and is encoded as a byte string. In JSON, the "id" value is a Base64 encoded byte string. In CBOR, the "id" type is byte string, and has label 8.

#### [4.](#) Client-RS Communication

Additionally to what specified in Section 4 of [\[I-D.ietf-ace-oscore-profile\]](#), the client also generates an identifier id1, unique in the sets of its own Recipient Ids, and posts it together with the token and nonce N1 to the RS. The RS also generates an identifier id2, unique in the sets of its own Recipient Ids, and uses it together with the rest of the input material to derive a security context. Both the nonces and identifiers are encoded as CBOR bstr if CBOR is used, and as Base64 string if JSON is used.

##### [4.1.](#) C-to-RS: POST to authz-info endpoint

Additionally to what specified in Section 4.1 of [\[I-D.ietf-ace-oscore-profile\]](#), the following applies.

The client generates its own Recipient Id for the OSCORE Security Context that it is establishing with the RS. By generating its own Recipient Id, the client makes sure that it does not collide with any of its Recipient Identifiers. The client posts it to the RS together with what is described in Section 4.1 of [\[I-D.ietf-ace-oscore-profile\]](#): the client includes the Recipient Id in the POST to authz-info request, as an `ace_client_recipientid` parameter, registered in [Section 7.2](#) and [Section 7.3](#).

When receiving the POST to authz-info request including the `ace_client_recipientid` parameter, the RS MUST set its own Sender Identifier to the value of the `ace_client_recipientid`. If the `ace_client_recipientid` parameter is not included in the request, the RS MUST stop processing the request and interrupt the Ace exchange, and MAY reply with a 4.00 (Bad Request) error message.

If the access token received contains either the `clientId` or `serverId`

parameters, the server SHOULD stop processing the message, and MAY reply with a 4.00 (Bad Request) error message.

Figure 9 shows an example of the request sent from the client to the RS, with payload in CBOR diagnostic notation without the tag and value abbreviations. The access token has been truncated for readability.

```
Header: POST (Code=0.02)
Uri-Host: "rs.example.com"
Uri-Path: "authz-info"
Content-Format: "application/ace+cbor"
Payload:
{
  "access_token": h'8343a1010aa2044c53 ...
(remainder of access token (CWT) omitted for brevity)',
  "nonce1": h'018a278f7faab55a',
  "ace_client_recipientid" : h'11'
}
```

Figure 9

#### [4.1.1.](#) The `ace_client_recipientid` Parameter

This parameter MUST be sent from the client to the RS, together with the access token and the nonce, if the ace profile used is `coap_oscore_2`. The parameter is encoded as a byte string for CBOR-based interactions, and as a string (Base64 encoded binary) for JSON-based interactions. This parameter is registered in [Section 7.2](#) and [Section 7.3](#).

#### [4.2.](#) RS-to-C: 2.01 (Created)

Additionally to what specified in Section 4.2 of [\[I-D.ietf-ace-oscore-profile\]](#), the following applies.

The RS generates its own Recipient Id for the OSCORE Security Context that it is establishing with the client. By generating its own Recipient Id, the RS makes sure that it does not collide with any of its Recipient Identifiers. The RS MUST ensure that `id2` is different

from the `ace_client_recipientid`. The RS sends it to the Client together with what is described in Section 4.2 of [\[I-D.ietf-ace-oscore-profile\]](#): the RS includes the Recipient Id in the 2.01 (Created) response, as a `ace_server_recipientid` parameter, as registered in [Section 7.2](#) and [Section 7.3](#).

When receiving the response including the `ace_server_recipientid` parameter, the Client MUST set its own Sender Identifier to the value of the `ace_server_recipientid` and discard any `ClientId` present in the access token. If the `ace_server_recipientid` parameter is not included in the response, the client MUST stop processing the response and interrupt the Ace exchange.

Figure 10 shows an example of the response sent from the RS to the client, with payload in CBOR diagnostic notation without the tag and value abbreviations.

```
Header: Created (Code=2.01)
Content-Format: "application/ace+cbor"
Payload:
{
  "nonce2": h'25a8991cd700ac01',
  "ace_server_recipientid" : h'22'
}
```

Figure 10

#### [4.2.1](#). The `ace_server_recipientid` Parameter

This parameter MUST be sent from the RS to the client, together with the access token and nonce, if the ace profile used is `coap_oscore_2`. The parameter is encoded as a byte string for CBOR-based interactions, and as a string (Base64 encoded binary) for JSON-based interactions. This parameter is registered in [Section 7.2](#) and [Section 7.3](#).

#### [4.3](#). OSCORE Setup

Differently than what defined in Section 4.3 of [\[I-D.ietf-ace-oscore-profile\]](#), the client and RS do not set the Sender ID and Recipient ID from the parameters received from the AS in [Section 3.2](#).

Instead, the client MUST set the Sender ID from the `ace_server_recipientid` received in [Section 4.2](#), and the Recipient ID from its own generated `ace_client_recipientid`. Conversely, the server MUST set the Sender ID from the `ace_client_recipientid` received in [Section 4.1](#), and the Recipient ID from its own generated `ace_server_recipientid`.

## [5.](#) Security Considerations

TODO

TBD: How do this profile and the v1 OSCORE profile work together? can the AS send a v1 profile + token and the c and RS try to run a v2? how does c react if it tries to run v2 and get reverted to v1? How do the 2 profiles work together?

## [6.](#) Privacy Considerations

TODO

## [7.](#) IANA Considerations

This document has the following actions for IANA.

### [7.1.](#) ACE Profile Registry

The following registration is done for the ACE Profile Registry following the procedure specified in section 8.8 of [\[I-D.ietf-ace-oauth-authz\]](#):

- o Name: `coap_oscore_2`
- o Description: Profile for using OSCORE to secure communication between constrained nodes using the Authentication and Authorization for Constrained Environments framework.
- o CBOR Value: TBD (value between 1 and 255)

- o Reference: `[[This specification]]`

## [7.2.](#) OAuth Parameters Registry

The following registrations are done for the OAuth ParametersRegistry following the procedure specified in [section 11.2 of \[RFC6749\]](#):

- o Parameter name: `ace_client_recipientid`
- o Parameter usage location: client request
- o Change Controller: IESG
- o Specification Document(s): `[[This specification]]`
  
- o Parameter name: `ace_server_recipientid`
- o Parameter usage location: token response
- o Change Controller: IESG
- o Specification Document(s): `[[This specification]]`

## [7.3.](#) OAuth Parameters CBOR Mappings Registry

The following registrations are done for the OAuth Parameters CBOR Mappings Registry following the procedure specified in section 8.9 of [\[I-D.ietf-ace-oauth-authz\]](#):

- \* Name: `ace_client_recipientid`
- \* CBOR Key: TBD (range -256 to 255)
- \* Value Type: byte string
- \* Reference: `\\[[This specification]\\]`

- \* Name: `ace_server_recipientid`
- \* CBOR Key: TBD (range -256 to 255)
- \* Value Type: byte string
- \* Reference: `\\[[This specification]\\]`

## [7.4.](#) OSCORE Security Context Parameters Registry

The following registration is done for the ACE Profile Registry following the procedure specified in section 9.4 of [\[I-D.ietf-ace-oscore-profile\]](#):

This registry will be populated by the values in Figure 8. The specification column for all of these entries will be this document.

## Acknowledgments

This document was started from comments and discussion with the following individuals: John Mattsson, Jim Schaad, Goeran Selander.

## 9. References

### 9.1. Normative References

- [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-35](#) (work in progress), June 2020.
- [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-11](#) (work in progress), June 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", [RFC 6749](#), DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/info/rfc6749>>.
- [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>>.

- [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>>.



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

## [9.2.](#) Informative References

[I-D.ietf-lake-edhoc]  
Selander, G., Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", [draft-ietf-lake-edhoc-01](#) (work in progress), August 2020.

### Author's Address

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

Email: [francesca.palombini@ericsson.com](mailto:francesca.palombini@ericsson.com)