ACE Working Group
Internet-Draft
Updates: 9200, 9201, 9202, 9203, 9431 (if approved)
Intended status: Standards Track
Expires: 23 April 2026

M. Tiloca
RISE AB
G. Selander
Ericsson AB

20 October 2025

Short Distribution Chain (SDC) Workflow and New OAuth Parameters for the Authentication and Authorization for Constrained Environments (ACE) Framework

draft-ietf-ace-workflow-and-params-06

Abstract

This document updates the Authentication and Authorization for Constrained Environments Framework (ACE, RFC 9200) as follows. (1) It defines the Short Distribution Chain (SDC) workflow that the authorization server (AS) can use for uploading an access token to a resource server on behalf of the client. (2) For the OAuth 2.0 token endpoint, it defines new parameters and encodings and it extends the semantics of the "ace_profile" parameter. (3) For the OAuth 2.0 authz-info endpoint, it defines a new parameter and its encoding. (4) It defines how the client and the AS can coordinate on the exchange of the client's and resource server's public authentication credentials, when those can be transported by value or identified by reference; this extends the semantics of the "rs_cnf" parameter for the OAuth 2.0 token endpoint, thus updating RFC 9201. (5) It extends the error handling at the AS, for which it defines a new error code. (6) It deprecates the original payload format of error responses conveying an error code, when CBOR is used to encode message payloads. For those responses, it defines a new payload format aligned with RFC 9290, thus updating in this respect also the profiles defined in RFC 9202, RFC 9203, and RFC 9431. (7) It amends two of the requirements on profiles of the framework.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-ace-workflow-and-params/.

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

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Authentication and Authorization for Constrained Environments (ACE) framework [RFC9200] defines an architecture to enforce access control for constrained devices. A client (C) requests an assertion of granted permissions from an authorization server (AS) in the form of an access token, then uploads the access token to the target resource server (RS), and finally accesses protected resources at the RS according to the permissions specified in the access token.

The framework has as main building blocks the OAuth 2.0 framework [RFC6749], the Constrained Application Protocol (CoAP) [RFC7252] for message transfer, Concise Binary Object Representation (CBOR) [RFC8949] for compact encoding, and CBOR Object Signing and Encryption (COSE) [RFC9052][RFC9053] for self-contained protection of access tokens. In addition, separate profile documents define in detail how the participants in the ACE architecture communicate, especially as to the security protocols that they use.

This document updates [RFC9200] as follows.

* It defines the Short Distribution Chain (SDC) workflow for the ACE framework (see Section 2), according to which the AS uploads the access token to the RS on behalf of C and then informs C about the outcome. The SDC workflow is especially convenient in deployments where the communication leg between C and the RS is constrained, but the communication leg between the AS and the RS is not.

  The SDC workflow has no ambition to replace the original workflow defined in [RFC9200]. The AS can use one workflow or the other depending, for example, on the specific RS for which an access token has been issued and the nature of the communication leg with that RS.

* It defines new parameters and encodings for the OAuth 2.0 token endpoint at the AS (see Section 3). These include:

  - "token_upload", used by C to inform the AS that it opts in to use the SDC workflow and by the AS to inform C about the outcome of the token uploading to the RS per the SDC workflow.

  - "token_hash", used by the AS to provide C with a token hash, which corresponds to an access token that the AS has issued for C and has successfully uploaded to the RS on behalf of C per the SDC workflow.

- "to_rs", used by C to provide the AS with information to relay to the RS, upon asking the AS to upload the access token to the RS per the SDC workflow. Its specific use with the OSCORE profile [RFC9203] is also described, as effectively enabling the use of the SDC workflow for that profile.

- "from_rs", used by the AS to provide C with information to relay from the RS, after the AS has successfully uploaded the access token to the RS per the SDC workflow. Its specific use with the OSCORE profile [RFC9203] is also described, as effectively enabling the use of SDC workflow for that profile.

- "rs_cnf2", used by the AS to provide C with the public keys of the RSs in the group-audience for which the access token is issued (see Section 6.9 of [RFC9200]).

- "audience2", used by the AS to provide C with the identifiers of the RSs in the group-audience for which the access token is issued.

- "anchor_cnf", used by the AS to provide C with the public keys of trust anchors, which C can use to validate the public key of an RS (e.g., as provided in the "rs_cnf" parameter defined in [RFC9201] or in the "rs_cnf2" parameter defined in this document).

- "token_series_id", used by the AS to provide C with the identifier of a token series and by C to ask the AS for a new access token in the same token series that dynamically updates access rights. A corresponding access token claim, namely "token_series_id", is also defined.

- "updated_rights", used by the AS to provide the RS with an indication that an access token uploaded per the SDC workflow is not the first one of a new token series, i.e., that the AS has issued the access token for dynamically updating the access rights of C.

* It extends the semantics of the "ace_profile" parameter for the OAuth 2.0 token endpoint at the authorization server defined in [RFC9200] (see Section 4).

* It defines how C and the AS can coordinate on the exchange of the client’s and resource server’s public authentication credentials, when those can be transported by value or identified by reference in the access token request and response (see Section 5).

This extends the semantics of the "rs_cnf" parameter for the OAuth 2.0 token endpoint defined in [RFC9201] and therefore updates [RFC9201].

* It extends the error handling at the AS, for which it defines a new error code that the AS can use for error responses sent to the client, after failing to verify the proof of possession of the client's private key when processing an access token request (see Section 6).

* It deprecates the original payload format of error responses that convey an error code, when CBOR is used to encode message payloads in the ACE framework. For such error responses, it defines a new payload format according to the problem-details format specified in [RFC9290] (see Section 7).

  In this respect, it also updates the profiles of the ACE framework defined in [RFC9202], [RFC9203], and [RFC9431].

* It amends two of the requirements on profiles of the ACE framework originally compiled in Appendix C of [RFC9200] (see Section 8).

1.1. Terminology

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

Readers are expected to be familiar with the terms and concepts described in the ACE framework for Authentication and Authorization [RFC9200][RFC9201], as well as with terms and concepts related to CBOR Web Tokens (CWTs) [RFC8392] and CWT Confirmation Methods [RFC8747].

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).

Readers are also expected to be familiar with the terms and concepts related to CoAP [RFC7252], Concise Data Definition Language (CDDL) [RFC8610], CBOR [RFC8949], JavaScript Object Notation (JSON) [RFC8259], and COSE [RFC9052][RFC9053].

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

Furthermore, this document uses the following terms.

* Token series: a set of access tokens, all of which are bound to the same proof-of-possession (PoP) key and are sequentially issued by the same AS for the same pair (client, audience) per the same profile of ACE. A token series ends when the latest access token of that token series becomes invalid (e.g., when it expires or gets revoked).

  Profiles of ACE can provide their extended and specialized definition, e.g., by further taking into account the public authentication credentials of C and the RS.

* Token hash: identifier of an access token, in binary format encoding. The token hash has no relation to other access token identifiers possibly used, such as the 'cti' (CWT ID) claim of CBOR Web Tokens (CWTs) [RFC8392].

CBOR [RFC8949] and CDDL [RFC8610] are used in this document. CDDL predefined type names, especially bstr for CBOR byte strings and tstr for CBOR text strings, are used extensively in this document.

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

In the CBOR diagnostic notation used in this document, constructs of the form e'SOME_NAME' are replaced by the value assigned to SOME_NAME in the CDDL model shown in Figure 11 of Appendix B. For example, {e'audience2' : ["rs1", "rs2"]} stands for {54 : ["rs1", "rs2"]}.

Note to RFC Editor: Please delete the paragraph immediately preceding this note. Also, in the CBOR diagnostic notation used in this document, please replace the constructs of the form e'SOME_NAME' with the value assigned to SOME_NAME in the CDDL model shown in Figure 11 of Appendix B. Finally, please delete this note.

2. The Short Distribution Chain (SDC) Workflow

As defined in Section 4 of [RFC9200], the ACE framework relies on its basic protocol workflow shown in Figure 1.

That is, the client first sends an access token request to the token endpoint at the AS (Step A), specifying permissions that it seeks to obtain for accessing protected resources at the RS, possibly together with information on its own public authentication credential.

Then, if the request has been successfully verified, authenticated, and authorized, the AS replies to the client (Step B), providing an access token and possibly additional parameters as access information including the actually granted permissions.

Finally, the client uploads the access token to the RS and, consistently with the permissions granted according to the access token, accesses a resource at the RS (Step C), which replies with the result of the resource access (Step F). Details about what protocol the client and the RS use to establish a secure association, mutually authenticate, and secure their communications are defined in the specific profile of ACE used, e.g., [RFC9202][RFC9203][RFC9431][I-D.ietf-ace-edhoc-oscore-profile][I-D.ietf-ace-group-oscore-profile].

Further interactions are possible between the AS and the RS, i.e., the exchange of an introspection request and response where the AS validates a previously issued access token for the RS (Steps D and E).

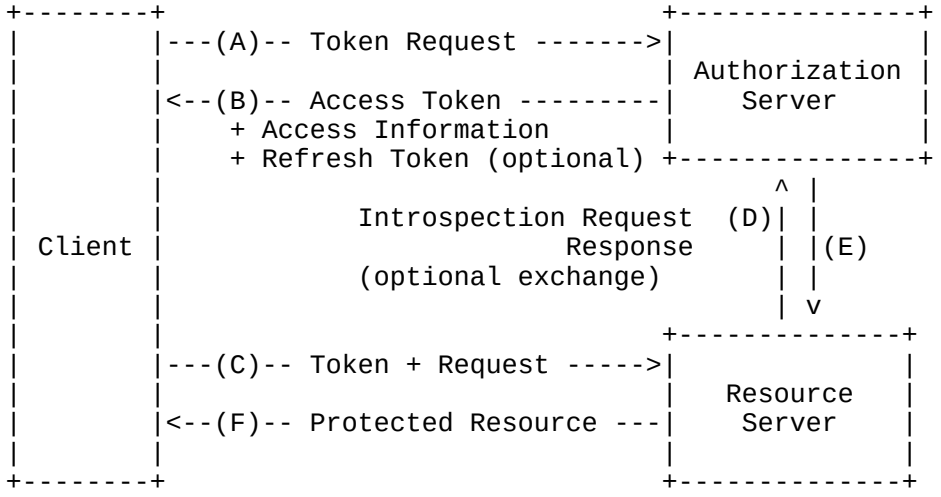

Figure 1: ACE Basic Protocol Workflow.

This section defines the alternative Short Distribution Chain (SDC) workflow shown in Figure 2, which MAY be supported by the AS. Unlike in the original workflow defined in [RFC9200], the AS uploads the access token to the RS on behalf of the client and then informs the client about the outcome.

If the token uploading has been successfully completed, the client typically does not need to obtain the access token from the AS altogether. That is, the client simply establishes a secure association with the RS (if that has not happened already) and then accesses protected resources at the RS, according to the permissions granted per the access token and specified by the AS as access information.

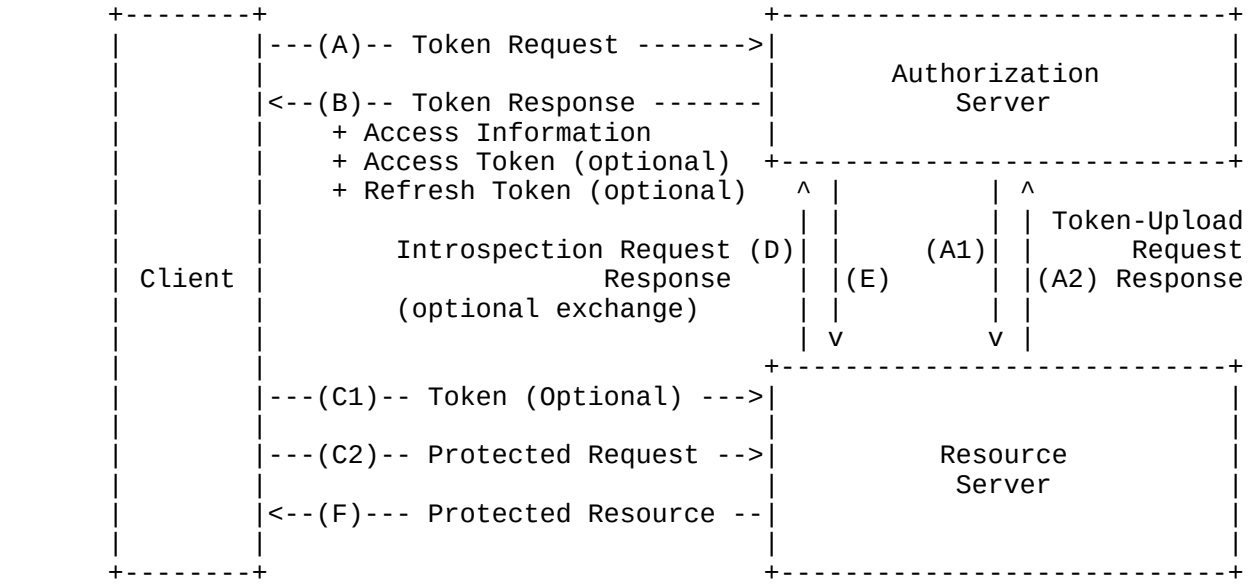

Figure 2: ACE Short Distribution Chain (SDC) Workflow.

More specifically, the SDC workflow consists of the following steps.

* Step A - Like in the original workflow, the client sends an access token request to the token endpoint at the AS, with the additional indication that it opts in to use the SDC workflow.

  As defined in Section 3.1, this information is conveyed to the AS by means of the "token_upload" parameter. The parameter also specifies what the AS has to return in the access token response at Step B, following a successful uploading of the access token from the AS to the RS.

* Step A1 - This new step consists of the AS uploading the access token to the RS, typically at the authz-info endpoint, just like the client does in the original workflow.

* Step A2 - This new step consists of the RS replying to the AS, following the uploading of the access token at Step A1.

* Step B - In the access token response, the AS tells the client that it has attempted to upload the access token to the RS, specifying the outcome of the token uploading based on the reply received from the RS at Step A2.

  As defined in Section 3.1, this information is conveyed to the client by means of the "token_upload" parameter included in the access token response. If the token uploading has failed, the access token response also includes the access token. Otherwise, the access token response includes information consistent with what was specified by the "token_upload" parameter of the access token request at Step A.

* Step C1 - This step occurs only if the token uploading from the AS has failed, and the AS has provided the client with the access token at Step B. In such a case, the client uploads the access token to the RS just like at Step C of the original workflow.

* Step C2 - The client attempts to access a protected resource at the RS, according to the permissions granted per the access token and specified by the AS as access information at Step B.

* Steps D, E, and F are as in the original workflow.

The SDC workflow has no ambition to replace the original workflow defined in [RFC9200]. The AS can use one workflow or the other depending, for example, on the specific RS for which the access token has been issued and the nature of the communication leg with that RS.

When using the SDC workflow, all the communications between the AS and the RS MUST be protected, consistent with Sections 5.8.4.3 and 6.5 of [RFC9200]. Unlike in the original workflow, this results in protecting also the uploading of the first access token in a token series, i.e., in addition to the uploading of the following access tokens in the token series for dynamically updating the access rights of the client.

The SDC workflow is also suitable for deployments where clients are not aware of details such as the need for access tokens to be issued by the AS and uploaded at the RS. Consistent with the intended access policies, the AS can be configured to automatically issue

access tokens for such clients and upload those access tokens to the RS. This means that such clients do not have to request for an access token to be issued in the first place. That is, they can immediately send requests to the RS for accessing its protected resources, in accordance with the access tokens already issued and uploaded by the AS.

2.1. Token Upload

When using the original workflow defined in [RFC9200], there are two typical cases concerning the upload of the access token to the RS from C.

* The first case considers the upload of the first access token in a token series. While details depend on the specific profile of ACE used, such an access token is typically uploaded through an unprotected POST request to the authz-info endpoint.

* The second case considers the upload of an access token that is not the first in its token series, i.e., the AS has issued the access token for dynamically updating the access rights of C. The intent is also for C and the RS to preserve the same secure communication association that they currently share and that is associated with the token series in question.

  Such an access token is uploaded through a POST request to the authz-info endpoint, which is protected by means of the secure communication association that is shared between C and the RS. If the new access token is accepted by the RS, the new access token supersedes the one currently stored for the token series in question, and it becomes associated with the corresponding secure communication association that is shared between C and the RS.

However, if the AS uploads the access token to the RS on behalf of C as per the SDC workflow, the upload of the access token always rely on a protected POST request from the AS to the authz-info endpoint. In particular, the request is protected with the secure communication association shared between the AS and the RS.

When receiving a POST request to the authz-info endpoint that is specifically protected through the secure communication association shared with the AS, the RS proceeds as follows. In particular, the RS leverages the explicit indication provided by the AS through the "updated_rights" parameter defined in Section 3.7, which is included in the request in the case of dynamic update of access rights.

* If the request does not include the "updated_rights" parameter, the RS processes the request and the access token therein like it would when receiving the request from C and according to the specific profile of ACE used. The access token in question is the first one in a token series.

* If the request includes the "updated_rights" parameter encoding the CBOR simple value true (0xf5), then the access token in question, namely T_NEW, is not the first one in its token series, i.e., it is meant to dynamically update the access rights of C, while preserving the same secure communication association that is shared between C and the RS.

  In this case, the RS uses lookup information specified within T_NEW to determine whether it stores an access token T_OLD associated with C and belonging to the same token series of T_NEW. Such lookup information includes an identifier of the token series to which both T_NEW and T_OLD belong, and it can further comprise additional information elements pertaining to the specific profile of ACE used (e.g., the authentication credential of C that is bound to the access tokens of the token series).

  By leveraging such lookup information, the RS checks whether it stores an access token T_OLD that belongs to the same token series of T_NEW, like it would when receiving the request from C and according to the specific profile of ACE used.

  If the RS is currently storing such an old access token T_OLD, then the RS MUST supersede T_OLD by replacing the corresponding authorization information with the one specified in the new access token T_NEW. After that, the RS MUST associate T_NEW with the same secure communication association with which T_OLD was associated.

  If the RS is not currently storing such an old access token T_OLD, then the RS MUST reject the POST request from the AS and MUST reply with an error response that has a response code equivalent to the CoAP code 5.00 (Internal Server Error).

The explicit indication provided by the "updated_rights" parameter prevents possible ambiguous situations, where the RS might erroneously believe that an access token is the first one in a new token series (e.g., following the deletion of stored access tokens due to memory limitations).

3. New Parameters

The rest of this section defines a number of additional parameters and encodings for the OAuth 2.0 token endpoint at the AS.

3.1. token_upload

This section defines the additional "token_upload" parameter. The parameter can be used in an access token request sent by C to the token endpoint at the AS, as well as in the successful access token response sent as reply by the AS.

* The "token_upload" parameter is OPTIONAL in an access token request. The presence of this parameter indicates that C opts in to use the SDC workflow defined in Section 2, whose actual use for uploading the issued access token to the RS is an exclusive prerogative of the AS.

  This parameter can take one of the following integer values. When the access token request is encoded in CBOR, those values are encoded as CBOR unsigned integers. The value of the parameter determines whether the follow-up successful access token response will have to include certain information, in case the AS has successfully uploaded the access token to the RS.

  - 0: The access token response will have to include neither the access token nor its corresponding token hash.

  - 1: The access token response will have to include the token hash corresponding to the access token, but not the access token.

  - 2: The access token response will have to include the access token, but not the corresponding token hash.

  If the AS supports the SDC workflow and the access token request includes the "token_upload" parameter with value 0, 1, or 2, then the AS MAY use the SDC workflow to upload the access token to the RS on behalf of C. Otherwise, following that access token request, the AS MUST NOT use the SDC workflow.

* The "token_upload" parameter is REQUIRED in a successful access token response with response code 2.01 (Created), if both the following conditions apply. Otherwise, the "token_upload" parameter MUST NOT be present.

  - The corresponding access token request included the "token_upload" parameter, with value 0, 1, or 2.

- The AS has attempted to upload the issued access token to the RS as per the SDC workflow, irrespective of the result of the token upload.

When the "token_upload" parameter is present in the access token response, it can take one of the following integer values. When the access token response is encoded in CBOR, those values are encoded as CBOR unsigned integers.

- If the token upload to the RS was not successful, then the "token_upload" parameter MUST encode the value 1.

  In this case, the access token response MUST include the "access_token" parameter specifying the issued access token.

- If the token upload at the RS was successful, then the "token_upload" parameter MUST encode the value 0.

  In this case, the access token response can include additional parameters as defined below, depending on the value of the "token_upload" parameter in the corresponding access token request.

  o If the "token_upload" parameter in the access token request specified the value 0, then the access token response MUST NOT include the "access_token" parameter and MUST NOT include the "token_hash" parameter defined in Section 3.2.

  o If the "token_upload" parameter in the access token request specified the value 1, then the access token response MUST NOT include the "access_token" parameter and MUST include the "token_hash" parameter defined in Section 3.2, specifying the hash corresponding to the issued access token and computed as defined in Section 3.2.

  o If the "token_upload" parameter in the access token request specified the value 2, then the access token response MUST include the "access_token" parameter specifying the issued access token and MUST NOT include the "token_hash" parameter defined in Section 3.2.

3.1.1. Examples

Figure 3 shows an example, with first an access token request from C to the AS and then an access token response from the AS to C, following the issue of an access token bound to a symmetric PoP key.

The access token request specifies the "token_upload" parameter with value 0. That is, C indicates that it requires neither the access token nor the corresponding token hash from the AS, in case the AS successfully uploads the access token to the RS.

The access token response specifies the "token_upload" parameter with value 0, which indicates that the AS has successfully uploaded the access token to the RS on behalf of C.

Consistent with the value of the "token_upload" parameter in the access token request, the access token response includes neither the access token nor its corresponding token hash. The access token response also includes the "cnf" parameter specifying the symmetric PoP key bound to the access token.

Access token request

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: 19 (application/ace+cbor)
Payload:
{
  / audience /  5 : "tempSensor4711",
  / scope /     9 : "read",
  e'token_upload' : 0
}
```

Access token response

```
Header: Created (Code=2.01)
Content-Format: 19 (application/ace+cbor)
Max-Age: 3560
Payload:
{
   e'token_upload' : 0,
  / expires_in / 2 : 3600,
  / cnf /        8 : {
    / COSE_Key / 1 : {
      / kty / 1 : 4 / Symmetric /,
      / kid / 2 : h'3d027833fc6267ce',
      / k /  -1 : h'73657373696f6e6b6579'
    }
  }
}
```

Figure 3: Example of Access Token Request-Response Exchange. Following a successful uploading of the access token from the AS to the RS, the access token response includes the "token_upload" parameter but not the access token, which is bound to a symmetric key and was uploaded to the RS by the AS.

Figure 4 shows another example, with first an access token request from C to the AS and then an access token response from the AS to C, also following the issue of an access token bound to a symmetric PoP key.

The access token request specifies the "token_upload" parameter with value 2. That is, C indicates that it requires the access token from the AS, even in case the AS successfully uploads the access token to the RS.

The access token response specifies the "token_upload" parameter with value 0, which indicates that the AS has successfully uploaded the access token to the RS on behalf of C.

Consistent with the value of the "token_upload" parameter in the access token request, the access token response includes the "access_token" parameter specifying the issued access token. The access token response also includes the "cnf" parameter specifying the symmetric PoP key bound to the access token.

```
Access token request

Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: 19 (application/ace+cbor)
Payload:
{
  / audience /  5 : "tempSensor4711",
  / scope /     9 : "read",
  e'token_upload' : 2
}


Access token response

Header: Created (Code=2.01)
Content-Format: 19 (application/ace+cbor)
Max-Age: 3560
Payload:
{
     e'token_upload' : 0,
  / access_token / 1 : h'd08343a1...4819',
    / (full CWT elided for brevity;
       CWT contains the symmetric PoP key in the "cnf" claim) /
  / expires_in /   2 : 3600,
  / cnf /          8 : {
    / COSE_Key / 1 : {
      / kty / 1 : 4 / Symmetric /,
      / kid / 2 : h'3d027833fc6267ce',
      / k /  -1 : h'73657373696f6e6b6579'
    }
  }
}
```

Figure 4: Example of Access Token Request-Response Exchange. Following a successful uploading of the access token from the AS to the RS, the access token response includes the "token_upload" parameter as well as the "access_token" parameter conveying the access token, which is bound to a symmetric key and was uploaded to the RS by the AS.

Figure 5 shows another example, with first an access token request from C to the AS and then an access token response from the AS to C, also following the issue of an access token bound to a symmetric PoP key.

The access token request specifies the "token_upload" parameter with value 0. That is, C indicates that it requires neither the access token nor the corresponding token hash from the AS, in case the AS successfully uploads the access token to the RS.

In this example, the access token response includes the "token_upload" parameter with value 1, which indicates that the AS has attempted and failed to upload the access token to the RS on behalf of C. The access token response also includes the "access_token" parameter specifying the issued access token, together with the "cnf" parameter specifying the symmetric PoP key bound to the access token.

Note that, even though the AS has failed to upload the access token to the RS, the response code 2.01 (Created) is used when replying to C, since the access token request as such has been successfully processed at the AS, with the following issue of the access token.

```
Access token request

Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: 19 (application/ace+cbor)
Payload:
{
  / audience /  5 : "tempSensor4711",
  / scope /     9 : "read",
  e'token_upload' : 0
}


Access token response

Header: Created (Code=2.01)
Content-Format: 19 (application/ace+cbor)
Max-Age: 3560
Payload:
{
     e'token_upload' : 1,
  / access_token / 1 : h'd08343a1...4819',
    / (full CWT elided for brevity;
       CWT contains the symmetric PoP key in the "cnf" claim) /
  / expires_in /   2 : 3600,
  / cnf /          8 : {
    / COSE_Key / 1 : {
      / kty / 1 : 4 / Symmetric /,
      / kid / 2 : h'3d027833fc6267ce',
      / k /  -1 : h'73657373696f6e6b6579'
    }
  }
}
```

Figure 5: Example of Access Token Request-Response Exchange. Following a failed uploading of the access token from the AS to the RS, the access token response includes the "token_upload" parameter with value 1 as well the "access_token" parameter conveying the access token bound to a symmetric key.

3.2. token_hash

This section defines the additional "token_hash" parameter. The parameter can be used in a successful access token response sent as reply by the AS to C.

The following refers to the base64url encoding without padding (see Section 5 of [RFC4648]) and denotes as "binary representation" of a text string the corresponding UTF-8 encoding [RFC3629], which is the implied charset used in JSON (see Section 8.1 of [RFC8259]).

The "token_hash" parameter is REQUIRED in a successful access token response with response code 2.01 (Created), if both the following conditions apply. Otherwise, the "token_hash" parameter MUST NOT be present.

* The corresponding access token request included the "token_upload" parameter with value 1.

* The access token response includes the "token_upload" parameter with value 0. That is, the AS has successfully uploaded the issued access token to the RS, as per the SDC workflow.

This parameter specifies the token hash corresponding to the access token issued by the AS and successfully uploaded to the RS on behalf of C. In particular:

* If the access token response is encoded in CBOR, then the "token_hash" parameter is a CBOR byte string, with value the token hash.

* If the access token response is encoded in JSON, then the "token_hash" parameter has as value the base64url-encoded text string that encodes the token hash.

The AS computes the token hash as defined in Section 3.2.1.

3.2.1. Computing the Token Hash

The AS computes the token hash over the value that the "access_token" parameter would have had in the same access token response, if it was included therein and specifying the access token.

In particular, the input HASH_INPUT over which the token hash is computed is determined as follows.

* If the access token response is encoded in CBOR, then:

  - BYTES denotes the value of the CBOR byte string that would be conveyed by the "access_token" parameter, if this was included in the access token response.

  - HASH_INPUT_TEXT is the base64url-encoded text string that encodes BYTES.

- HASH_INPUT is the binary representation of HASH_INPUT_TEXT.

* If the access token response is encoded in JSON, then HASH_INPUT is the binary representation of the text string conveyed by the "access_token" parameter, if this was included in the access token response.

Once determined HASH_INPUT as defined above, a hash value of HASH_INPUT is generated as per Section 6 of [RFC6920]. The resulting output in binary format is used as the token hash. Note that the used binary format embeds the identifier of the used hash function in the first byte of the computed token hash.

The specific hash function used MUST be collision resistant on byte strings and MUST be selected from the "Named Information Hash Algorithm Registry" [IANA.Hash.Algorithms]. Consistent with the compliance requirements in Section 2 of [RFC6920], the hash function sha-256 as specified in [SHA-256] is mandatory to implement.

The computation of token hashes defined above is aligned with that specified for the computation of token hashes in Section 4 of [RFC9770], where they are used as identifiers of revoked access tokens. Therefore, given a hash algorithm and an access token, the AS computes the same corresponding token hash in either case.

If the AS supports the method specified in [RFC9770], then the AS MUST use the same hash algorithm for computing both the token hashes to include in the "token_hash" parameter and the token hashes computed per that method to identify revoked access tokens.

3.2.2. Example

Figure 6 shows an example, with first an access token request from C to the AS and then an access token response from the AS to C, following the issue of an access token bound to a symmetric PoP key.

The access token request specifies the "token_upload" parameter with value 1. That is, C indicates that it requires the token hash corresponding to the access token from the AS, in case the AS successfully uploads the access token to the RS.

The access token response specifies the "token_upload" parameter with value 0, which indicates that the AS has successfully uploaded the access token to the RS on behalf of C.

Consistent with the value of the "token_upload" parameter in the access token request, the access token response includes the "token_hash" parameter, which specifies the token hash corresponding

to the issued access token. The access token response also includes the "cnf" parameter specifying the symmetric PoP key bound to the access token.

Access token request

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: 19 (application/ace+cbor)
Payload:
{
  / audience /  5 : "tempSensor4711",
  / scope /     9 : "read",
  e'token_upload' : 1
}
```

Access token response

```
Header: Created (Code=2.01)
Content-Format: 19 (application/ace+cbor)
Max-Age: 3560
Payload:
{
   e'token_upload' : 0,
     e'token_hash' : h'0153269057e12fe2b74ba07c892560a2d7
                       53877eb62ff44d5a19002530ed97ffe4',
  / expires_in / 2 : 3600,
  / cnf /        8 : {
    / COSE_Key / 1 : {
      / kty / 1 : 4 / Symmetric /,
      / kid / 2 : h'3d027833fc6267ce',
      / k /  -1 : h'73657373696f6e6b6579'
    }
  }
}
```

Figure 6: Example of Access Token Request-Response Exchange. Following a successful uploading of the access token from the AS to the RS, the access token response includes the "token_upload" parameter as well as the "token_hash" parameter. The "token_hash" parameter conveys the token hash corresponding to the issued access token, which is bound to a symmetric key and was uploaded to the RS by the AS.

3.3. to_rs and from_rs

The rest of this section defines the additional parameters "to_rs" (see Section 3.3.1) and "from_rs" (see Section 3.3.2).

The "to_rs" parameter can be used in an access token request sent by C to the token endpoint at the AS. The "from_rs" parameter can be used in an access token response sent by the AS, in reply to a request to the token endpoint from C.

The semantics and encoding of the information specified in the two parameters depend on the specific profile of ACE and application used.

For instance, when using the OSCORE profile of ACE [RFC9203], C and RS can use the two parameters to exchange the expected nonces and identifiers via the AS. Section 3.3.3 describes how the two parameters are used when the OSCORE profile of ACE is used, thereby effectively enabling the use of the SDC workflow for that profile.

3.3.1. to_rs

The "to_rs" parameter is OPTIONAL in an access token request. The presence of this parameter indicates that:

* C wishes the AS to relay the information specified therein to the RS, when the AS uploads the issued access token to the RS per the SDC workflow defined in Section 2; and

* C wishes the AS to relay information received from the RS to C, after having successfully uploaded the access token to the RS per the SDC workflow defined in Section 2.

This parameter MUST NOT be present if the "token_upload" parameter defined in Section 3.1 is not present in the access token request. Also, this parameter MUST NOT be present if the requested access token is not the first one of a new token series, i.e., if C is asking the AS for a new access token in the same token series that dynamically updates access rights.

If C wishes that the AS relays information from the RS after successfully uploading the access token but C does not have any information to be relayed to the RS, then this parameter MUST encode the CBOR simple value null (0xf6).

Otherwise, this parameter specifies the information that C wishes the AS to relay to the RS, when uploading the access token to the RS on behalf of C.

Such information consists in what C would provide to the RS in addition to the access token, by sending a POST request to the authz-info endpoint in case the original workflow was used.

When composing the parameter "to_rs", C considers the same information and MUST wrap it in a data structure STRUCT, by using the same method that it employs when using the original workflow. For example, with reference to the OSCORE profile of ACE [RFC9203], such a method composes STRUCT as a CBOR map, which has to be sent as payload of a request with Content-Format "application/ace+cbor".

After that, C builds a CBOR byte string STR, whose value is the binary representation of STRUCT.

If and only if the request to the authz-info endpoint has to be sent with a Content-Format ct different from the one employed in the profile of ACE used, then C MUST tag the CBOR byte string STR (see Section 3.4 of [RFC8949]). The tag number TN MUST be the one associated with the Content-Format ct and is determined according to the technique described in Appendix B of [RFC9277]. For example, the Content-Format "application/ace+cbor" has Content-Format ID 19 and therefore has 1668546836 as its associated tag number.

When the access token request is encoded in CBOR, the value of the "to_rs" parameter is the (tagged) CBOR byte string STR.

When the access token request is encoded in JSON, the value of the "to_rs" parameter is a text string, which encodes the binary representation of the (tagged) CBOR byte string STR in base64url without padding (see Section 5 of [RFC4648]).

If the "to_rs" parameter is present and specifies a value different from the CBOR simple value null (0xf6), the AS proceeds as follows when composing the POST request to send to the authz-info endpoint on behalf of C:

1. The AS sets the Content-Format of the request to be either:

   * the one employed in the profile of ACE used, if the CBOR byte string STR conveyed by the "to_rs" parameter is not tagged; or, otherwise

   * the one associated with the tag number TN of the tagged CBOR byte string STR conveyed by the "to_rs" parameter.

2. The AS retrieves the structure STRUCT from STR and extends STRUCT by adding the issued access token, consistently with the profile of ACE used and the Content-Format determined at the previous step.

   For example, when using the OSCORE profile of ACE [RFC9203] and thus the Content-Format to use is "application/ace+cbor", STRUCT is a CBOR map and the access token is included therein as an entry that has: as map key, 1 encoded as a CBOR integer; as map value, a CBOR byte string whose value is the binary representation of the access token.

Clearly, performing the two steps above requires the AS to understand the structure STRUCT and its semantics. In turn, such an understanding builds on the AS supporting the profile of ACE used and the Content-Format to use as determined at Step 1. In the expected cases, the AS is realistically able to perform the two steps. If the AS finds itself unable to perform the two steps, then the AS would simply not upload the access token to the RS on behalf of C. In such a case, the AS replies to C with a successful access token response to C, which includes the "access_token" parameter specifying the issued access token and does not include the "token_upload" parameter (see Section 3.1).

Tagging the CBOR byte string as defined above ensures that the AS can relay the information specified in the "to_rs" parameter as intended by C, i.e., by sending to the authz-info endpoint a POST request that has the correct Content-Format and conveys the correct payload.

As a case in point, using the DTLS profile of ACE [RFC9202] typically results in a POST request to the authz-info endpoint with Content-Format "application/cwt", as per Section 5.10 of [RFC9200]. However, the DTLS profile of ACE can be combined with application profiles of [RFC9594]. In such a case, the POST request might convey both the access token and additional parameters from C (e.g., "sign_info" defined in Section 3.3 of [RFC9594]), which requires the request to have Content-Format "application/ace+cbor" (see Section 3.3 of [RFC9594]).

3.3.2. from_rs

The "from_rs" parameter is OPTIONAL in an access token response. The presence of this parameter indicates that the AS is relaying the information specified therein to C, which the AS has received from the RS after having successfully uploaded the access token to the RS per the SDC workflow defined in Section 2.

This parameter MUST be present if and only if both the following conditions apply:

* The "token_upload" parameter defined in Section 3.1 is present with value 0 in the access token response.

* The "to_rs" parameter was present in the access token request corresponding to the access token response.

This parameter specifies the information that the AS is relaying to C from the RS, following the successful upload of the access token to the RS on behalf of C.

Such information consists in what C would receive in a successful response from the authz-info endpoint, if the access token was uploaded per the original workflow.

When composing the parameter "from_rs", the AS builds a CBOR byte string STR as follows:

* If the successful response from the authz-info endpoint does not include a payload, then STR is the empty CBOR byte string (0x40); otherwise

* The value of STR is the payload of the successful response from the authz-info endpoint.

  If and only if the response specifies a given Content-Format ct (i.e., by means of the CoAP Content-Format Option), then the AS MUST tag the CBOR byte string STR (see Section 3.4 of [RFC8949]). The tag number TN MUST be the one associated with the Content-Format ct and is determined according to the technique described in Appendix B of [RFC9277].

When the access token response is encoded in CBOR, the value of the "from_rs" parameter is the (tagged) CBOR byte string STR.

When the access token response is encoded in JSON, the value of the "from_rs" parameter is a text string, which encodes the binary representation of the (tagged) CBOR byte string STR in base64url without padding (see Section 5 of [RFC4648]).

When C receives from the AS the successful access token response specifying the "token_upload" parameter with value 0, C retrieves from the "from_rs" parameter the information relayed by the AS, just like when retrieving that information from a 2.01 (Created) response with Content-Format ct that C receives from the RS when using the original workflow.

In particular, if the CBOR byte string STR is not the empty CBOR byte string (0x40), C considers as the Content-Format ct either:

* the one employed in the profile of ACE used, if the CBOR byte string STR conveyed by the "from_rs" parameter is not tagged; or, otherwise

* the one associated with the tag number TN of the tagged CBOR byte string STR conveyed by the "from_rs" parameter.

3.3.3. Use with the OSCORE Profile

This section describes how to use the parameters "to_rs" and "from_rs" when the OSCORE profile of ACE [RFC9203] is used.

Within the access token request from C to the AS, the value of the "to_rs" parameter is a CBOR byte string, whose value is the binary representation of a CBOR map C_MAP composed of two fields:

* A field with the CBOR unsigned integer 40 as map key and with value the nonce N1 generated by C, encoded a CBOR byte string (see Section 4.1 of [RFC9203]).

* A field with the CBOR unsigned integer 43 as map key and with value the Recipient ID ID1 generated by C, encoded as a CBOR byte string (see Section 4.1 of [RFC9203]).

When building the POST request for uploading the access token to the authz-info endpoint at the RS, the AS sets the Content-Format of the request to "application/ace+cbor" and composes the request payload as specified in Section 4.1 of [RFC9203]. In particular, the CBOR map specified as payload includes:

* The "access_token" field, with value the access token to upload encoded as a CBOR byte string and with the CBOR unsigned integer 1 as map key.

* The "nonce1" field, with value the same CBOR byte string specified by the field of C_MAP that has the CBOR unsigned integer 40 as map key.

* The "ace_client_recipientid" field, with value the same CBOR byte string specified by the field of C_MAP that has the CBOR unsigned integer 43 as map key.

If the upload of the access token to the RS from the AS is successful, the RS replies to the AS with a 2.01 (Created) response, which has Content-Format "application/ace+cbor" and whose payload is a CBOR map RS_MAP that includes:

* The "nonce2" field, with value the nonce N2 generated by the RS, encoded as a CBOR byte string (see Section 4.2 of [RFC9203]).

* The "ace_server_recipientid" field, with value the Recipient ID ID2 generated by the RS, encoded as a CBOR byte string (see Section 4.2 of [RFC9203]).

Within the access token response from the AS to C, the value of the "from_rs" parameter is a CBOR byte string, whose value is the binary representation of a CBOR map composed of two elements:

* A field with the CBOR unsigned integer 42 as map key and with value the same CBOR byte string specified by the "nonce2" field of RS_MAP.

* A field with the CBOR unsigned integer 44 as map key and with value the same CBOR byte string specified by the "ace_server_recipientid" field of RS_MAP.

When C receives from the AS the successful access token response specifying the "token_upload" parameter with value 0, C retrieves the nonce N2 and the Recipient ID ID2 from the "from_rs" parameter, just like when retrieving those from a 2.01 (Created) response received from the RS when using the original workflow.

Figure 7 shows an example where the OSCORE profile is used, with first an access token request from C to the AS and then an access token response from the AS to C, following the issue of an access token bound to a symmetric PoP key.

The access token request specifies the "token_upload" parameter with value 0. That is, C indicates that it requires neither the access token nor the corresponding token hash from the AS, in case the AS successfully uploads the access token to the RS. Also, the access token request includes the "to_rs" parameter, specifying the values of N1 = 0x018a278f7faab55a and ID1 = 0x1645 intended to the RS.

The access token response specifies the "token_upload" parameter with value 0, which indicates that the AS has successfully uploaded the access token to the RS on behalf of C.

Consistent with the value of the "token_upload" parameter in the access token request, the access token response includes neither the access token nor its corresponding token hash. The access token response also includes the "cnf" parameter specifying the symmetric PoP key bound to the access token, as an OSCORE_Input_Material object (see Section 3.2.1 of [RFC9203]). Also, the access token response includes the "from_rs" parameter, specifying the values of N2 = 0x25a8991cd700ac01 and ID2 = 0x0000 received from the RS and intended to C.

Access token request

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: 19 (application/ace+cbor)
Payload:
{
  / audience /  5 : "tempSensor4711",
  / scope /     9 : "read",
  e'token_upload' : 0,
         e'to_rs' : h'a2182848018a278f7faab55a182b421645'
}
```

Access token response

```
Header: Created (Code=2.01)
Content-Format: 19 (application/ace+cbor)
Max-Age: 3560
Payload:
{
     e'token_upload' : 0,
          e'from_rs' : h'a2182a4825a8991cd700ac01182c420000',
  / ace_profile / 38 : 2 / coap_oscore /,
    / expires_in / 2 : 3600,
    / cnf /        8 : {
      / osc / 4 : {
        / id / 0 : h'01',
        / ms / 2 : h'f9af838368e353e78888e1426bd94e6f'
      }
    }
}
```

Figure 7: Example of Access Token Request-Response Exchange where the OSCORE Profile is Used. Following a successful uploading of the access token from the AS to the RS, the access token response includes the "token_upload" parameter but not the access token, which is bound to a symmetric key and was uploaded to the RS by the AS. C and the RS exchange N1, ID1, N2, and ID2 via the AS by means of the parameters "to_rs" and "from_rs"

3.4. rs_cnf2 and audience2

This section defines the additional parameters "rs_cnf2" and "audience2" for an access token response, sent by the AS in reply to a request to the token endpoint from C.

* The "rs_cnf2" parameter is OPTIONAL if the token type is "pop", asymmetric keys are used, and the access token is issued for an audience that includes multiple RSs (i.e., a group-audience, see Section 6.9 of [RFC9200]). Otherwise, the "rs_cnf2" parameter MUST NOT be present.

  This parameter specifies information about the public keys used by the RSs of a group-audience for authenticating themselves to C. It is used in case the binding between the public keys and the corresponding RS identities are not established through other means. If this parameter is absent, either the RSs in the group-audience do not use a public key, or the AS knows that the RSs can authenticate themselves to C without additional information.

  If present, this parameter MUST encode a non-empty CBOR array of N elements, where N is the number of RSs in the group-audience for which the access token is issued. Each element of the CBOR array specifies the public key of one RS in the group-audience, and MUST follow the syntax and semantics of the "cnf" claim either from Section 3.1 of [RFC8747] for CBOR-based interactions, or from Section 3.1 of [RFC7800] for JSON-based interactions. It is not required that all the elements of the CBOR array rely on the same confirmation method.

  Any of the public keys may be provided together with information such as the public key algorithm and use (e.g., specified by means of the parameters "alg" and "key_ops" in a COSE_Key structure). If such information is specified, a client MUST NOT use a public key that is incompatible with the profile of ACE used or with the PoP algorithm according to that information. An RS MUST reject a proof of possession that relies on such a key and MUST reply with a response code equivalent to the CoAP code 4.00 (Bad Request).

* The "audience2" parameter is OPTIONAL and specifies the identifiers of the RSs in the group-audience for which the access token is issued.

  If present, this parameter MUST encode a non-empty CBOR array of N elements, where N is the number of RSs in the group-audience for which the access token is issued. Each element of the CBOR array in the "audience2" parameter MUST be a CBOR text string, with value the identifier of one RS in the group-audience.

  The element of the CBOR array referring to an RS in the group-audience SHOULD have the same value that would be used to identify that RS through the "audience" parameter of an access token request to the AS (see Section 5.8.1 of [RFC9200]) and of an access token response from the AS (see Section 5.8.2 of [RFC9200]), when requesting and issuing an access token for that individual RS.

  The "audience2" parameter is REQUIRED if the "rs_cnf2" parameter is present. In such a case, the i-th element of the CBOR array in the "audience2" parameter MUST be the identifier of the RS whose public key is specified as the i-th element of the CBOR array in the "rs_cnf2" parameter.

3.4.1. Example

Figure 8 shows an example of access token response from the AS to C, following the issue of an access token for a group-audience composed of two RSs "rs1" and "rs2" and bound to C's public key as asymmetric PoP key. The access token response includes the access token as well as the parameters "audience2" and "rs_cnf2". These specify the public key of the two RSs as intended recipients of the access token and the identifiers of those two RSs, respectively.

Access token response

```
Header: Created (Code=2.01)
Content-Format: 19 (application/ace+cbor)
Max-Age: 3600
Payload:
{
  / access_token / 1 : b64'SlAV32hk...12',
    / (full CWT elided for brevity;
       CWT contains the client's RPK in the "cnf" claim) /
  / expires_in /   2 : 3600,
       e'audience2' : ["rs1", "rs2"],
         e'rs_cnf2' : [
           {
             / COSE_Key / 1 : {
               / kty /  1 : 2 / EC2 /,
               / crv / -1 : 1 / P-256 /,
               / x /   -2 : h'bbc34960526ea4d32e940cad2a234148
                             ddc21791a12afbcbac93622046dd44f0',
               / y /   -3 : h'4519e257236b2a0ce2023f0931f1f386
                             ca7afda64fcde0108c224c51eabf6072'
             }
           },
           {
             / COSE_Key / 1 : {
               / kty /  1 : 2 / EC2 /,
               / crv / -1 : 1 / P-256 /,
               / x /   -2 : h'ac75e9ece3e50bfc8ed6039988952240
                             5c47bf16df96660a41298cb4307f7eb6',
               / y /   -3 : h'6e5de611388a4b8a8211334ac7d37ecb
                             52a387d257e6db3c2a93df21ff3affc8'
             }
           }
         ]
}
```

Figure 8: Example of access token response with an access token bound to an asymmetric key, using the parameters "audience2" and "rs_cnf2".

3.5. anchor_cnf

This section defines the additional "anchor_cnf" parameter for an access token response, sent by the AS in reply to a request to the token endpoint from C.

The "anchor_cnf" parameter is OPTIONAL if the token type is "pop" and asymmetric keys are used. Otherwise, the "anchor_cnf" parameter MUST NOT be present.

This parameter specifies information about the public keys of trust anchors, which C can use to validate the public key of the RS/RSs included in the audience for which the access token is issued. This parameter can be used when the access token is issued for an audience including one RS or multiple RSs.

If this parameter is absent, either the RS/RSs in the audience do not use a public key, or the AS knows that C can validate the public key of such RS/RSs without additional information (e.g., C has already obtained the required public keys of the involved trust anchors from the AS or through other means).

If present, this parameter MUST encode a non-empty CBOR array that MUST be treated as a set, i.e., the order of its elements has no meaning. Each element of the CBOR array specifies the public key of one trust anchor, which can be used to validate the public key of at least one RS included in the audience for which the access token is issued. Each element of the CBOR array MUST follow the syntax and semantics of the "cnf" claim either from Section 3.1 of [RFC8747] for CBOR-based interactions, or from Section 3.1 of [RFC7800] for JSON-based interactions. It is not required that all the elements of the CBOR array rely on the same confirmation method.

Any of the public keys conveyed in the "anchor_cnf" parameter may be provided together with information such as the public key algorithm and use (e.g., specified by means of the parameters "alg" and "key_ops" in a COSE_Key structure). If such information is specified, a client MUST NOT use a public key that is incompatible with the profile of ACE used or with the public keys to validate and the way to validate those.

The presence of this parameter does not require that the access token response also includes the "rs_cnf" parameter defined in [RFC9201] or the "rs_cnf2" parameter defined in Section 3.4 of this document. That is, C may be able to obtain the public keys of the RS/RSs for which the access token is issued through other means.

When the access token response includes both the "anchor_cnf" parameter and the "audience2" parameter defined in Section 3.4, then C MUST make sure that a public key PK_RS is associated with an RS identified by an element of "audience2", before using any of the public keys specified in "anchor_cnf" to validate PK_RS.

When the access token response includes the "anchor_cnf" parameter but not the "audience2" parameter, then C can use any of the public keys specified in "anchor_cnf" to validate the public key PK_RS of any RS in the targeted audience. This allows C to use the access token with an RS that is deployed later on as part of the same audience, which is particularly useful in the case of a group-audience.

3.5.1. Example

Figure 9 shows an example of access token response from the AS to C, following the issue of an access token for a group-audience and bound to C's public key as asymmetric PoP key.

The identifier of the group-audience was specified by the "audience" parameter of the access token request to the AS, is specified by the "aud" claim of the issued access token, and is not repeated in the access token response from the AS.

The access token response includes the "anchor_cnf" parameter. This specifies the public key of a trust anchor that C can use to validate the public keys of any RS with which the access token is going to be used. By means of the CWT confirmation method "x5chain" defined in [I-D.ietf-ace-edhoc-oscore-profile], the public key of the trust anchor is here conveyed within an X.509 certificate [RFC5280] used as public authentication credential for that trust anchor.

```
Access token response

Header: Created (Code=2.01)
Content-Format: 19 (application/ace+cbor)
Max-Age: 3600
Payload:
{
  / access_token / 1 : b64'SlAV32hk...12',
    / (full CWT elided for brevity;
       CWT contains the client's RPK in the "cnf" claim) /
  / expires_in /   2 : 3600,
       e'anchor_cnf' : [
         {
           e'x5chain' : h'308201363081dea003020102020301f50d30
                          0a06082a8648ce3d0403023016311430120 6
                          035504030c0b52464320746573742043413 0
                          1e170d3230303130313030303030305a170d
                          3231303230323030303030305a3022312030
                          1e06035504030c1730312d32332d34352d46
                          462d46452d36372d38392d41423059301306
                          072a8648ce3d020106082a8648ce3d030107
                          03420004b1216ab96e5b3b3340f5bdf02e69
                          3f16213a04525ed44450b1019c2dfd3838ab
                          ac4e14d86c0983ed5e9eef2448c6861cc406
                          547177e6026030d051f7792ac206a30f300d
                          300b0603551d0f040403020780300a06082a
                          8648ce3d0403020347003044022044 5d798c
                          90e7f500dc747a654cec6cfa6f037276e14e
                          52ed07fc16294c84660d02205a33985dfbd4
                          bfdd6d4acf3804c3d46ebf3b7fa62640674f
                          c0354fa056dbaea6'
         }
       ]
}
```

Figure 9: Example of Access Token Response with an access token bound to an asymmetric key, using the "anchor_cnf" parameter.

3.6. token_series_id

This section defines the additional "token_series_id" parameter. The parameter can be used in an access token request sent by C to the token endpoint at the AS as well as in the successful access token response sent as reply by the AS.

* The "token_series_id" parameter is OPTIONAL in an access token request. The presence of this parameter indicates that C wishes to obtain a new access token for dynamically updating its access

rights. That is, the new access token is intended to be the next one in an active token series and to supersede the latest access token in that token series. This parameter MUST NOT be present if the requested access token is the first one of a new token series.

If present, this parameter specifies the identifier of the token series that the new access token is intended to extend. The identifier does not change throughout the lifetime of the token series and was provided to C in the successful access token response that the AS sent when issuing the first access token in that token series. When the access token request is encoded in CBOR, the value of this parameter is encoded as a CBOR byte string.

* The "token_series_id" parameter is OPTIONAL in an access token response. This parameter MUST NOT be present if the issued access token is not the first one of the token series it belongs to.

  If present, this parameter specifies the identifier of the token series to which the issued access token belongs. When the access token response is encoded in CBOR, the value of this parameter is encoded as a CBOR byte string.

If the AS relies on the "token_series_id" parameter to exchange the identifier of token series with clients, then the following applies.

* The value assigned to the identifier of a token series MUST be associated with all the access tokens issued by the AS for that token series and MUST be selected from a pool that the AS exclusively controls.

  In particular, the triple (TS_ID, C, AUD) MUST uniquely identify a token series and its corresponding access tokens, where TS_ID is the identifier of the token series, while C and AUD are the client and the audience for which the access token is issued, respectively. The AS MUST take into account both ongoing and ended token series for selecting a new TS_ID that complies with the above requirements.

  Note that the ACE profile is not part of the triple, hence the requirement spans across all the ACE profiles that the AS and its registered clients/RSs support.

* An issued access token that belongs to a token series MUST include the identifier of that token series. This allows the RS to identify the latest access token in the token series to be superseded by the issued access token.

In particular, each of such access tokens MUST include a claim specifying the identifier of the token series to which the access token belongs. When CWTs are used as access tokens, this information MUST be transported in the "token_series_id" claim registered in Section 10.4.

If a profile of ACE relies on a construct that uses different parameters/claims to transport the identifier of a token series, then the new "token_series_id" parameter and "token_series_id" claim MUST NOT be used when using that profile.

For example, a number of parameters/claims are already used to transport information that acts de facto as identifier of token series, in the PSK mode of the DTLS profile [RFC9202], in the OSCORE profile [RFC9203], and in the EDHOC and OSCORE profile [I-D.ietf-ace-edhoc-oscore-profile].

3.7. updated_rights

This section defines the additional "updated_rights" parameter. The parameter can be used in a POST request sent by the AS to the authz-info endpoint, when the AS uploads an access token to the RS per the SDC workflow defined in Section 2. The "updated_rights" parameter MUST NOT be included in the POST request to the authz-info endpoint sent by C per the original workflow defined in [RFC9200].

In the POST request from the AS, the "updated_rights" parameter is REQUIRED if the uploaded access token is not the first one of a new token series, i.e., if the AS has issued the access token for dynamically updating the access rights of C. Otherwise, the "updated_rights" parameter MUST NOT be present.

When including the "updated_rights" parameter, the POST request MUST have Content-Format "application/ace+cbor" and its payload MUST be formatted as a CBOR map. In particular, the CBOR map MUST include the "updated_rights" parameter encoding the CBOR simple value true (0xf5), together with the "access_token" parameter specifying the access token. The CBOR map MAY include additional parameters, according to the specific profile of ACE used.

Note that this request deviates from the POST request defined in [RFC9200], although such a deviation can already occur in some profiles of ACE (e.g., see Section 4.1 of [RFC9203]) or in application profiles of [RFC9594].

When the RS receives a protected POST request to the authz-info endpoint from the AS and the request does not convey the "updated_rights" parameter, the RS is ensured that the access token conveyed in the request is the first one of a new token series.

When the RS receives a protected POST request to the authz-info endpoint from the AS and the request conveys the "updated_rights" parameter encoding the CBOR simple value true (0xf5), the RS is ensured that the access token conveyed in the request is not the first one of a new token series.

Taking advantage of such explicit indication requires the RS to support the Content-Format "application/ace+cbor" and the "updated_rights" parameter.

If the RS does not support the Content-Format "application/ace+cbor", the RS rejects the POST request from the AS and replies with an error response that has a response code equivalent to the CoAP code 4.15 (Unsupported Content-Format). Note that, irrespective of the profile of ACE used, the RS supports the Content-Format "application/ace+cbor" if it implements token introspection (see Section 5.9 of [RFC9200]).

If the RS supports the Content-Format "application/ace+cbor" but does not support the "updated_rights" parameter, the RS MUST reject the POST request from the AS and MUST reply with an error response that has a response code equivalent to the CoAP code 4.00 (Bad Request).

In case a POST request to the authz-info endpoint conveys the "updated_rights" parameter, the RS supports the parameter, and any of the following conditions applies, the RS MUST reject the request and MUST reply with an error response that has a response code equivalent to the CoAP code 4.00 (Bad Request):

* The request is not protected.

* The AS is not successfully verified as the originator of the request.

* The "updated_rights" parameter does not encode the CBOR simple value true (0xf5).

If the AS receives an error response from the RS, the AS replies to C with a successful access token response, which includes the "access_token" parameter specifying the issued access token and includes the "token_upload" parameter encoding the value 1 (see Section 3.1).

The processing of the POST request to the authz-info endpoint from the AS is further described in Section 2.1.

4. Updated "ace_profile" Parameter

This section extends the semantics of the "ace_profile" parameter defined in [RFC9200] for the OAuth 2.0 token endpoint at the authorization server.

In addition to what is specified in Sections 5.8.1, 5.8.2, and 5.8.4.3 of [RFC9200], the following applies.

* When sending an access token request to the token endpoint at the AS (see Section 5.8.1 of [RFC9200]), C MAY include the "ace_profile" parameter, specifying the identifier of the profile that C wishes to use towards the RS.

* If the AS receives an access token request that includes the "ace_profile" parameter specifying the identifier of a profile, then the AS proceeds as follows.

  In case the AS does not issue access tokens per the profile specified in the access token request, or C and the RS do not share that profile, then the AS MUST reject the request and MUST reply with an error response (see Section 5.8.3 of [RFC9200]). The error response MUST have a response code equivalent to the CoAP code 4.00 (Bad Request) and MUST include the error code "incompatible_ace_profiles".

  In case the AS issues an access token to C, the access token MUST be per the profile whose identifier was specified by the "ace_profile" parameter in the access token request.

  In case the AS replies to C with a successful access token response (see Section 5.8.2 of [RFC9200]), then the response MAY include the "ace_profile" parameter. If it is included in the access token response, the "ace_profile" parameter MUST encode the same profile identifier that was specified by the "ace_profile" parameter of the corresponding access token request.

5. Coordinating on the Exchange of Public Authentication Credentials

In some profiles of ACE, it is possible for C and the RS to use public authentication credentials. Depending on the specific profile, the access token request and response exchanged between C and the AS can specify those authentication credentials as transported by value or instead identified by reference. For instance, this is the case in the EDHOC and OSCORE profile

[I-D.ietf-ace-edhoc-oscore-profile] and in the DTLS profile [RFC9202] as extended in [I-D.ietf-ace-authcred-dtls-profile].

At some point, the AS (C) might become unable to use a credential identifier as a reference for accessing the authentication credential of C (of the RS) obtained earlier, e.g., due to having deleted the credential from its local storage. This can prevent the AS (C) from successfully processing an incoming access token request (response) that specifies the authentication credential of C (of the RS) as identified by reference. Ultimately, this can prevent the AS from issuing an access token and C from securely accessing protected resources at the RS.

Conversely, unbeknown to the AS, C might already be storing the authentication credential of the RS when sending the access token request. In such a situation, the AS would specify the authentication credential of the RS by value in the access token response. However, it would be sufficient for C that the response specified the credential of the RS as identified by reference, or even that the response omitted the credential altogether.

In order to allow C and the AS to coordinate on the exchange of the authentication credentials of C and the RS, the rest of this section defines:

* How the AS can instruct C to specify its public authentication credential by value in the "req_cnf" parameter of an access token request (see Section 5.1).

* How C can instruct the AS to specify the public authentication credential(s) of the RS(s) by value or by reference in the "rs_cnf" or "rs_cnf2" parameter of an access token response (see Section 5.2), or instead to omit the credential(s) from the access token response.

5.1. Instructing C on How to Provide its Authentication Credential

When the AS receives an access token request and this includes the "req_cnf" parameter identifying the public authentication credential of C by reference, it might happen that the AS is not able to access the credential by using the specified reference.

In such a case, the AS MUST reject the request and MUST reply with an error response (see Section 5.8.3 of [RFC9200]). The error response MUST have a response code equivalent to the CoAP code 5.00 (Internal Server Error) and MUST include the error code "unknown_credential_referenced". The error code and its CBOR abbreviation are registered in Section 10.5 and Section 10.6, respectively.

After receiving such an error response, C can send a new access token request, where the "req_cnf" parameter specifies the authentication credential of C by value.

5.2. Instructing the AS on How to Provide the RS's Authentication Credential

When C receives an access token response and this includes the "req_cnf" or "req_cnf2" parameter identifying the authentication credential(s) of the RS(s) by reference, it might happen that C is not able to access the authentication credential(s) by using the specified reference(s).

Conversely, if the response includes the "rs_cnf" or "rs_cnf2" parameter specifying the authentication credential(s) of the RS(s) by value, it might happen that C has already been storing those credential(s), unbeknown to the AS. In fact, it would have been sufficient that the "rs_cnf" or "rs_cnf2" parameter identified the credential(s) by reference, or that neither parameter was included in the response.

The following extends the semantics of the "rs_cnf" parameter defined in [RFC9201], so that C can include the "rs_cnf" parameter in an access token request. When doing so, C instructs the AS about whether and how the successful access token response should specify the authentication credential(s) of the RS(s) belonging to the targeted audience.

Per its extended semantics, the "rs_cnf" parameter is also OPTIONAL to include in an access token request for requesting the first access token in a token series, if the token type is "pop" and asymmetric keys are used. Otherwise, the parameter MUST NOT be included in an access token request.

When C includes the "rs_cnf" parameter in an access token request, the parameter MUST have one of the following encodings.

* If the parameter specifies the CBOR simple value true (0xf5), then it instructs the AS to include in the successful access token response the "rs_cnf" or "rs_cnf2" parameter, specifying the authentication credential(s) of the RS(s) by value.

  In the successful access token response, each pertaining authentication credential MUST be specified by value.

* If the parameter specifies the CBOR simple value false (0xf4), then it instructs the AS to include in the successful access token response the "rs_cnf" or "rs_cnf2" parameter, specifying the authentication credential(s) of the RS(s) by reference.

  In the successful access token response, each pertaining authentication credential MUST be specified by reference, or alternatively by value only in case that was not possible.

* If the parameter specifies the CBOR simple value null (0xf6), then it instructs the AS to omit the "rs_cnf" and "rs_cnf2" parameters from the successful access token response.

  In the successful access token response, the "rs_cnf" and "rs_cnf2" parameters MUST NOT be included.

If the AS is not able to comply in the first two cases above, then the AS MUST reject the request and MUST reply with an error response. The error response MUST have a response code equivalent to the CoAP code 5.00 (Internal Server Error).

Irrespective of what "rs_cnf" specifies in the access token request, C MUST rely on the authentication credential(s) specified by the parameter "rs_cnf" or "rs_cnf2" in the access token response, as those that are used by the RS(s) to authenticate.

If C does not currently store the authentication credential(s) of the RS(s), then C MUST NOT include the "rs_cnf" parameter specifying the CBOR simple value null (0xf6) in an access token request.

6. Failed Verification of Proof of Possession at the AS

When sending an access token request to the AS (see Section 5.8.1 of [RFC9200]), a client can include the "req_cnf" parameter defined in Section 3.1 of [RFC9201] in order to provide the AS with a specific PoP key to bind to the requested access token.

Typically, the PoP key in question is the client's public key. In such a case, as per Section 3.1 of [RFC9201], the AS has to verify proof of possession of the client's private key, i.e., that the client is indeed in possession of the private key corresponding to the public key conveyed in the "req_cnf" parameter.

The AS might have previously achieved proof of possession of the private key in question, e.g., from previous interactions with the client or through out-of-band means. Alternatively, a profile of ACE might define how the AS verifies a PoP evidence that the client computes and provides to the AS by means of a parameter included in the access token request (e.g., see [I-D.ietf-ace-group-oscore-profile]).

Irrespective of the method used, if the AS fails to verify the proof of possession of the client's private key, then the AS MUST reject the access token request and MUST reply with an error response (see Section 5.8.3 of [RFC9200]). The error response MUST have a response code equivalent to the CoAP code 4.00 (Bad Request) and MUST include the error code "failed_pop_verification". The error code and its CBOR abbreviation are registered in Section 10.5 and Section 10.6, respectively.

7. Updated Payload Format of Error Responses

This section deprecates the original payload format of error responses conveying an error code, when CBOR is used to encode message payloads in the ACE framework. That format is referred to, e.g., when defining the error responses of Sections 5.8.3 and 5.9.3 of [RFC9200].

Also, this section defines a new payload format that allows such error responses to convey an error code together with further error-specific information, according to the problem-details format specified in [RFC9290].

Such error responses MUST have Content-Format set to "application/concise-problem-details+cbor". The payload of these error responses MUST be a CBOR map specifying a Concise Problem Details data item (see Section 2 of [RFC9290]). The CBOR map is formatted as follows:

* It MUST include the Custom Problem Detail entry "ace-error" registered in Section 10.7 of this document.

  This entry is formatted as a CBOR map including only one field, namely "error-code". The map key for the "error-code" field is the CBOR unsigned integer with value 0. The value of the "error-code" field is a CBOR integer specifying the error code associated

with the occurred error. This value is taken from the "CBOR Value" column of the "OAuth Error Code CBOR Mappings" registry [ACE.OAuth.Error.Code.CBOR.Mappings].

The new payload format MUST use the "error-code" field in order to convey the same information that the original payload format conveys through the "error" parameter (see, e.g., Sections 5.8.3 and 5.9.3 of [RFC9200]).

The CDDL notation [RFC8610] of the "ace-error" entry is given below.

```
ace-error = {
  &(error-code: 0) => int
}
```

* It MAY include further Standard Problem Detail entries or Custom Problem Detail entries (see [RFC9290]). The following Standard Problem Detail entries are of particular relevance for the ACE framework.

  - "detail" (map key -2): its value is a CBOR text string that specifies a human-readable diagnostic description of the occurred error (see Section 2 of [RFC9290]).

    The diagnostic text is intended for software engineers as well as for device and network operators in order to aid in debugging and provide context for possible intervention. The diagnostic message SHOULD be logged by the sender of the error response. The "detail" entry is unlikely to be relevant in an unattended setup where human intervention is not expected.

    The new payload format MUST use the Standard Problem Detail entry "detail" in order to convey the same information that the original payload format conveys through the "error_description" parameter (see, e.g., Sections 5.8.3 and 5.9.3 of [RFC9200]).

  - "instance" (map key -3): its value is a URI reference identifying the specific occurrence of the error (see Section 2 of [RFC9290]).

    The new payload format MUST use the Standard Problem Detail entry "instance" in order to convey the same information that the original payload format conveys through the "error_uri" parameter (see, e.g., Sections 5.8.3 and 5.9.3 of [RFC9200]).

An example of an error response using the problem-details format is shown in Figure 10.

```
Header: Bad Request (Code=4.00)
Content-Format: 257 (application/concise-problem-details+cbor)
Payload:
{
  / title /  -1 : "Incompatible ACE profile",
  / detail / -2 : "The RS supports only the OSCORE profile",
    e'ace-error': {
      / error_code / 0: 8 / incompatible_ace_profiles /
    }
}
```

Figure 10: Example of Error Response with Problem Details.

When the ACE framework is used with CBOR for encoding message payloads, the following applies.

* It is RECOMMENDED that authorization servers, clients, and resource servers support the payload format defined in this section.

* Authorization servers, clients, and resource servers that support the payload format defined in this section MUST use it when composing an outgoing error response that conveys an error code.

8. Updated Requirements on Profiles of ACE

Appendix C of [RFC9200] compiles a list of requirements on the profiles of ACE. This document amends two of those requirements as follows.

The text of the fifth requirement

> Specify the security protocol the client and RS must use to protect their communication (e.g., OSCORE or DTLS). This must provide encryption and integrity and replay protection (Section 5.8.4.3).

is replaced by the following text:

> Specify the security protocol the client and RS must use to protect their communication (e.g., OSCORE or DTLS). In combination with the used communication protocol, this must provide encryption, integrity and replay protection, and a binding between requests and responses (Section 5.8.4.3 and Section 6.5).

The text of the tenth requirement

| Specify the communication and security protocol for interactions
| between the client and AS. This must provide encryption,
| integrity protection, replay protection, and a binding between
| requests and responses (Sections 5 and 5.8).

is replaced by the following text:

| Specify the communication and security protocol for interactions
| between the client and AS. The combined use of those protocols
| must provide encryption, integrity protection, replay protection,
| and a binding between requests and responses (Sections 5 and 5.8).

At the time of writing, all the profiles of ACE that are published as RFC (i.e., [RFC9202][RFC9203][RFC9431]) already comply with the two updated requirements as formulated above.

9. Security Considerations

The same security considerations from the ACE framework for Authentication and Authorization [RFC9200] apply to this document, together with those from the specific profile of ACE used, e.g., [RFC 9202][RFC9203][RFC9431][I-D.ietf-ace-edhoc-oscore-profile][I-D.ietf-ace-group-oscore-profile][RFC9431].

When using the problem-details format defined in [RFC9290] for error responses, then the privacy and security considerations from Sections 4 and 5 of [RFC9290] also apply.

Editor’s note: add more security considerations.

10. IANA Considerations

This document has the following actions for IANA.

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

10.1. OAuth Parameters Registry

IANA is asked to add the following entries to the "OAuth Parameters" registry within the "OAuth Parameters" registry group.

* Name: token_upload

* Parameter Usage Location: token request, token response

* Change Controller: IETF

* Reference: [RFC-XXXX]

* Name: token_hash
* Parameter Usage Location: token response
* Change Controller: IETF
* Reference: [RFC-XXXX]

* Name: to_rs
* Parameter Usage Location: token request
* Change Controller: IETF
* Reference: [RFC-XXXX]

* Name: from_rs
* Parameter Usage Location: token response
* Change Controller: IETF
* Reference: [RFC-XXXX]

* Name: rs_cnf2
* Parameter Usage Location: token response
* Change Controller: IETF
* Reference: [RFC-XXXX]

* Name: audience2
* Parameter Usage Location: token response

* Change Controller: IETF
* Reference: [RFC-XXXX]

* Name: anchor_cnf
* Parameter Usage Location: token response
* Change Controller: IETF
* Reference: [RFC-XXXX]

* Name: token_series_id
* Parameter Usage Location: token request, token response
* Change Controller: IETF
* Reference: [RFC-XXXX]

* Name: updated_rights
* Parameter Usage Location: as-rs request
* Change Controller: IETF
* Reference: [RFC-XXXX]

In the same registry, IANA is asked to update the entries for the following OAuth parameters identified by their name, so that the content of the "Parameter Usage Location" column and of the "Reference" column is as below:

* rs_cnf
  - Parameter Usage Location: token request, token response
  - Reference: [RFC9201, Section 5][RFC-XXXX, Section 5.2]
* nonce1

- Parameter Usage Location: client-rs request, as-rs request
- Reference: [RFC9203][RFC-XXXX]

* nonce2
  - Parameter Usage Location: rs-client response, rs-as response
  - Reference: [RFC9203][RFC-XXXX]

* ace_client_recipientid
  - Parameter Usage Location: client-rs request, as-rs request
  - Reference: [RFC9203][RFC-XXXX]

* ace_server_recipientid
  - Parameter Usage Location: rs-client response, rs-as response
  - Reference: [RFC9203][RFC-XXXX]

* sign_info
  - Parameter Usage Location: client-rs request, rs-client response, as-rs request, rs-as response
  - Reference: [RFC9594][RFC-XXXX]

* kdcchallenge
  - Parameter Usage Location: rs-client response, rs-as response
  - Reference: [RFC9594][RFC-XXXX]

10.2. OAuth Parameters CBOR Mappings Registry

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

* Name: token_upload
* CBOR Key: TBD (value between 1 and 255)
* Value Type: unsigned integer

* Reference: [RFC-XXXX]
* Original Specification: [RFC-XXXX]

* Name: token_hash
* CBOR Key: TBD (value between 1 and 255)
* Value Type: unsigned integer
* Reference: [RFC-XXXX]
* Original Specification: [RFC-XXXX]

* Name: to_rs
* CBOR Key: TBD (value between 1 and 255)
* Value Type: Null or byte string or #6.<uint>(bstr)
* Reference: [RFC-XXXX]
* Original Specification: [RFC-XXXX]

* Name: from_rs
* CBOR Key: TBD (value between 1 and 255)
* Value Type: byte string or #6.<uint>(bstr)
* Reference: [RFC-XXXX]
* Original Specification: [RFC-XXXX]

* Name: rs_cnf2
* CBOR Key: TBD (value between 1 and 255)
* Value Type: array

* Reference: [RFC-XXXX]
* Original Specification: [RFC-XXXX]

* Name: audience2
* CBOR Key: TBD (value between 1 and 255)
* Value Type: array
* Reference: [RFC-XXXX]
* Original Specification: [RFC-XXXX]

* Name: anchor_cnf
* CBOR Key: TBD (value between 1 and 255)
* Value Type: array
* Reference: [RFC-XXXX]
* Original Specification: [RFC-XXXX]

* Name: token_series_id
* CBOR Key: TBD (value between 1 and 255)
* Value Type: byte string
* Reference: [RFC-XXXX]
* Original Specification: [RFC-XXXX]

* Name: updated_rights
* CBOR Key: TBD (value between 1 and 255)
* Value Type: True

* Reference: [RFC-XXXX]

* Original Specification: [RFC-XXXX]

In the same registry, IANA is asked to update the entry for the OAuth parameter with name "rs_cnf", so that the content of the "Value Type" column and of the "Reference" column is as below:

* Value Type: True or False or Null or map

* Reference: [RFC9201, Section 3.2][RFC-XXXX, Section 5.2]

10.3. JSON Web Token Claims Registry

IANA is asked to add the following entry to the "JSON Web Token Claims" registry within the "JSON Web Token (JWT)" registry group, following the procedure specified in [RFC7519].

* Claim Name: token_series_id

* Claim Description: The identifier of a token series

* Change Controller: IETF

* Reference: [RFC-XXXX]

10.4. CBOR Web Token (CWT) Claims Registry

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

* Claim Name: token_series_id

* Claim Description: The identifier of a token series

* JWT Claim Name: token_series_id

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

* Claim Value Type: byte string

* Change Controller: IETF

* Reference: Section 3.6 of [RFC-XXXX]

10.5. OAuth Extensions Error Registry

IANA is asked to add the following entries to the "OAuth Extensions Error Registry" within the "OAuth Parameters" registry group.

* Name: unknown_credential_referenced
* Usage Location: token error response
* Protocol Extension: [RFC-XXXX]
* Change Controller: IETF
* Reference: Section 5 of [RFC-XXXX]

* Name: failed_pop_verification
* Usage Location: token error response
* Protocol Extension: [RFC-XXXX]
* Change Controller: IETF
* Reference: Section 6 of [RFC-XXXX]

10.6. OAuth Error Code CBOR Mappings Registry

IANA is asked to add the following entries to the "OAuth Error Code CBOR Mappings" registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group.

* Name: unknown_credential_referenced
* CBOR Value: TBD (value between 1 and 255)
* Reference: [RFC-XXXX]
* Original Specification: [RFC-XXXX]

* Name: failed_pop_verification
* CBOR Value: TBD (value between 1 and 255)
* Reference: [RFC-XXXX]

* Original Specification: [RFC-XXXX]

10.7. Custom Problem Detail Keys Registry

IANA is asked to register the following entry in the "Custom Problem Detail Keys" registry within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

* Key Value: TBD (value between 0 and 23)

* Name: ace-error

* Brief Description: Carry ACE [RFC9200] problem details in a Concise Problem Details data item.

* Change Controller: IETF

* Reference: Section 7 of [RFC-XXXX]

11. References

11.1. Normative References

[ACE.OAuth.Error.Code.CBOR.Mappings]
IANA, "OAuth Error Code CBOR Mappings", <https://www.iana.org/assignments/ace/ace.xhtml#oauth-error-code-cbor-mappings>.

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

[IANA.Hash.Algorithms]
IANA, "Named Information Hash Algorithm Registry", <https://www.iana.org/assignments/named-information/named-information.xhtml>.

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/rfc/rfc3629>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

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

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

[RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keranen, A., and P. Hallam-Baker, "Naming Things with Hashes", RFC 6920, DOI 10.17487/RFC6920, April 2013, <https://www.rfc-editor.org/rfc/rfc6920>.

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

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/rfc/rfc7800>.

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

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

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

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

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

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

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

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

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

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

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

[RFC9202] Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)", RFC 9202, DOI 10.17487/RFC9202, August 2022, <https://www.rfc-editor.org/rfc/rfc9202>.

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

[RFC9277] Richardson, M. and C. Bormann, "On Stable Storage for Items in Concise Binary Object Representation (CBOR)", RFC 9277, DOI 10.17487/RFC9277, August 2022, <https://www.rfc-editor.org/rfc/rfc9277>.

[RFC9290] Fossati, T. and C. Bormann, "Concise Problem Details for Constrained Application Protocol (CoAP) APIs", RFC 9290, DOI 10.17487/RFC9290, October 2022, <https://www.rfc-editor.org/rfc/rfc9290>.

[RFC9430] Bergmann, O., Preu Mattsson, J., and G. Selander, "Extension of the Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE) to Transport Layer Security (TLS)", RFC 9430, DOI 10.17487/RFC9430, July 2023, <https://www.rfc-editor.org/rfc/rfc9430>.

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

[RFC9770] Tiloca, M., Palombini, F., Echeverria, S., and G. Lewis, "Notification of Revoked Access Tokens in the Authentication and Authorization for Constrained Environments (ACE) Framework", RFC 9770, DOI 10.17487/RFC9770, June 2025, <https://www.rfc-editor.org/rfc/rfc9770>.

[SHA-256] NIST, "Secure Hash Standard", NIST FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4 , August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

11.2. Informative References

[I-D.ietf-ace-authcred-dtls-profile] Tiloca, M. and J. P. Mattsson, "Additional Formats of Authentication Credentials for the Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)", Work in Progress, Internet-Draft, draft-ietf-ace-authcred-dtls-profile-02, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-authcred-dtls-profile-02>.

[I-D.ietf-ace-group-oscore-profile] Tiloca, M., Hglund, R., and F. Palombini, "The Group Object Security for Constrained RESTful Environments (Group OSCORE) Profile of the Authentication and Authorization for Constrained Environments (ACE) Framework", Work in Progress, Internet-Draft, draft-ietf-ace-group-oscore-profile-05, 3 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-group-oscore-profile-05>.

[RFC9594] Palombini, F. and M. Tiloca, "Key Provisioning for Group Communication Using Authentication and Authorization for Constrained Environments (ACE)", RFC 9594, DOI 10.17487/RFC9594, September 2024, <https://www.rfc-editor.org/rfc/rfc9594>.

Appendix A. Benefits for ACE Profiles

For any profile of ACE, the following holds.

* The SDC workflow defined in Section 2 is effectively possible to use. This is beneficial for deployments where the communication leg between C and the RS is constrained, but the communication leg between the AS and RS is not.

* When the SDC workflow is used, the "token_upload" parameter defined in Section 3.1 is used:

  - To inform the AS about C opting in to use the SDC workflow.

- To request the AS that the follow-up successful access token response includes certain information, if the AS has successfully uploaded the access token to the RS.

- To inform C that the AS has attempted to upload the issued access token to the RS, specifying whether the uploading has succeeded or failed.

* When the SDC workflow is used, it remains possible for C to always obtain the issued access token from the AS.

  That is, by specifying the value 2 for the "token_upload" parameter in the access token request, C will ensure to receive the access token from the AS, even if the AS successfully uploads the access token to the RS on behalf of C.

  This is useful in profiles of ACE where C can re-upload the same access token to the RS by itself, e.g., in order to perform a key update like defined for the OSCORE profile [RFC9203].

A.1. DTLS Profile

When the RPK mode of the DTLS profile is used (see Section 3.2 of [RFC9202]), it becomes possible for the AS to effectively issue an access token intended to an audience that includes multiple RSs.

This is enabled by the parameters "rs_cnf2" and "audience2" defined in Section 3.4 as well as by the "anchor_cnf" parameter defined in Section 3.5. This seamlessly applies also if the profile uses Transport Layer Security (TLS) [RFC8446] as defined in [RFC9430].

A.2. EDHOC and OSCORE Profile

When the EDHOC and OSCORE profile is used [I-D.ietf-ace-edhoc-oscore-profile], it becomes possible for the AS to effectively issue an access token intended to an audience that includes multiple RSs.

This is enabled by the parameters "rs_cnf2" and "audience2" defined in Section 3.4 as well as by the "anchor_cnf" parameter defined in Section 3.5.

Appendix B. CDDL Model

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

```
   ; OAuth Parameters CBOR Mappings
   token_upload = 49
   token_hash = 50
   to_rs = 51
   from_rs = 52
   rs_cnf2 = 53
   audience2 = 54
   anchor_cnf = 55
   token_series_id_param = 56

   ; CBOR Web Token (CWT) Claims
   token_series_id_claim = 42

   ; CWT Confirmation Methods
   x5chain = 24

   ; Custom Problem Detail Keys Registry
   ace-error = 2
```

Figure 11: CDDL model

Appendix C. Document Updates

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

C.1. Version -05 to -06

* Defined dynamic update of access rights in the SDC workflow.

* Defined the new "updated_rights" parameter.

* Clarified practical requirements at the AS for processing the "to_rs" parameter.

* IANA considerations: update in the "Parameter Usage Location" column for some entries of the "OAuth Parameters" registry.

* Adjusted abbreviations in the CDDL model to avoid collisions.

* Removed appendix with placeholder ideas.

* Editorial fixes and improvements.

C.2. Version -04 to -05

* Error handling and error code for failed PoP verification at the AS.

* Extended definition of the parameters "to_rs" and "from_rs".
* Fixes and presentation improvements in the IANA considerations.
* Updated references.
* Revised order of some sections.
* Editorial fixes and improvements.

C.3. Version -03 to -04

* Updated document title.
* Defined name for the new workflow.
* Improved definition of "token series".
* Revised note on the new workflow suitable for "unaware" clients.
* Revised criterion for the AS to choose a token series identifier.
* Extended semantics of the "ace_profile" parameter.
* Specified means for C and the AS to coordinate on the exchange of public authentication credentials.
* Removed content on bidirectional access control.
* Suggested value ranges for codepoints to register.
* Editorial fixes and improvements.

C.4. Version -02 to -03

* Defined parameter and claim "token_series_id".
* Defined parameters "to_rs" and "from_rs".
* Defined use of "to_rs" and "from_rs" in the OSCORE profile.
* Lowercase use of "client", "resource server", and "authorization server".
* Fixed naming of parameters/claims for audience.
* Split elision and comments in the examples in CBOR Diagnostic Notation.

* SHA-256 is mandatory to implement for computing token hashes.
* Fixes in the IANA considerations.
* Removed (parts of) appendices that are not needed anymore.
* Clarifications and editorial improvements.

C.5. Version -01 to -02

* CBOR diagnostic notation uses placeholders from a CDDL model.
* Note on the new workflow supporting also non-ACE clients.
* Revised semantics of the "token_upload" parameter.
* Defined the new "token_hash" parameter.
* First definition of bidirectional access control through a single access token.
* Revised and extended considerations and next steps in appendices.
* Clarifications and editorial improvements.

C.6. Version -00 to -01

* Definition of the "token series" moved to the "Terminology" section.
* Clarifications and fixes on using parameters in messages.
* Amended two of the requirements on profiles of the framework.
* The client has to opt-in for using the new workflow.
* Parameter "token_uploaded" renamed to "token_upload".
* Updated format of error response payload to use RFC 9290.
* Security considerations inherited from other documents.
* Editorial fixes and improvements.

Acknowledgments

The authors sincerely thank Christian Amsss, Rikard Hglund, and Dave Robin for their comments and feedback.

This work was supported by the Sweden's Innovation Agency VINNOVA within the EUREKA CELTIC-NEXT project CYPRESS; and by the H2020 project SIFIS-Home (Grant agreement 952652).

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden
Email: marco.tiloca@ri.se

Gran Selander
Ericsson AB
Torshamnsgatan 23
SE-16440 Stockholm Kista
Sweden
Email: goran.selander@ericsson.com