Recap

Set of updates to RFC 9200

1. Define an alternative workflow for uploading the access token
   – The AS uploads the access token to the RS, on behalf of C
   – Preferable if the C-RS communication leg is constrained, while the AS-RS leg is not

2. Define additional OAuth parameters to use in ACE
   – To enable the alternative workflow above
   – To effectively enable the issue of an access token for a group-audience
   – ...

3. Amend two requirements on transport profiles of ACE

4. Deprecate the original payload format of error responses
   – Instead, use the problem-details format from RFC 9290
Updates in v -02

› Clarifications and editorial improvements

› Notation in the examples
  – The CBOR diagnostic notation uses the construct e'' (*) to import values from the CDDL model in Appendix C
  › e'SOME_NAME' is replaced by the value assigned to SOME_NAME in the CDDL model
  › For example, {e'aud_2' : ["rs1", "rs2"]} stands for {50 : ["rs1", "rs2"]}.

/* Access Token Response */
Header: Created (Code=2.01)
Content-Format: 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'
    }
  }
}
/* Access Token Response */
Header: Created (Code=2.01)
Content-Format: application/ace+cbor
Max-Age: 3560
Payload:
{
  e'token_upload': 0,
  e'token_hash': h'0153269057e12fe2b74ba07c892560a2d7
                  53877be62ff4d5a19002530ed97ffe4',
  / expires_in / 2 : 3600,
  / cnf / 8 : {
    / COSE_Key / 1 : {
      / kty / 1 : 4 : Symmetric /
      / kid / 2 : h'3d027833fc6267ce',
      / k / -1 : h'73657373696f6e6b6579'
    }
  }
}

Updates in v -02

Alternative workflow

› Section 2 – Secure communication between AS and RS
  – Stated as explicit requirement, aligned with RFC 9200
  – The AS protects the uploading of every access token, i.e., also the first one in a token series

› Section 2 – Remark: usable also by non-ACE Clients (Thanks, Dave Robin!)
  – If properly configured, the AS just issues access tokens and uploads them to the RS
  – As properly configured, Clients just interact with the RS, without asking for an access token

› Related parameters
  – Revised semantics of “token_upload”
  – Defined the new “token_hash”
Updates in v -02

› “token_upload” – Value type changed to unsigned integer
  – C indicates its wish for obtaining the access token, or a token hash, or neither
  – The AS indicates the outcome of the token upload to the RS

› “token_upload” in the Access Token Request (C to AS)
  – If present: the AS is requested to use the alternative workflow (the final decision remains on the AS)
  – 0: if the token upload to the RS succeeds, C wants neither the access token nor the token hash
  – 1: if the token upload to the RS succeeds, C wants the token hash
  – 2: if the token upload to the RS succeeds, C wants the access token

› “token_upload” in the Access Token Response (AS to C)
  – Must be present if and only if the AS used the alternative workflow, irrespective of the outcome
  – 0: the token upload to the RS succeeded
    › The Response includes (access token / token hash / neither), depending on “token_upload” in the Request
  – 1: the token upload to the RS failed
    › The Response includes the access token
Updates in v -02

› “token_hash” – The value type is byte string
  – Only intended for the Access Token Response (AS to C)

› Only used if all the following points hold:
  – In the Access Token request, C specified “token_upload” = 1
  – The AS actually used the alternative workflow
  – The upload of the access token to the RS was successful

› Parameter value
  – The AS computes it from the issued access token, like it does when using the service defined in [1]
  – If the AS provides the service in [1], then the AS must use the same hash algorithm used for that service

› When using the alternative workflow, why would C want to obtain ...
  – ... the token hash? ➔ To be still able to use the service defined in [1]
  – ... the access token? ➔ To possibly re-upload the access token to the RS by itself (e.g., see RFC 9203)
Updates in v -02

Alternative workflow – Examples of **successful** token uploading from the AS

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

/ Access Token Response /
Header: Created (Code=2.01)
Content-Format: application/ace+cbor
Max-Age: 3568
Payload:
  
  e'token_upload' : 0,
  
  e'token_hash' : h'0153269057e12fe2b74ba07c892560a2d753877eb62ff44d5a1902530ed97ffe4',
  / expires_in / 2 : 3600,
  / cnf / 8 : {  / COSE_Key / 1 : {  / kty / 1 : 4 / Symmetric /,
    / kid / 2 : h'3d027833fc6267ce',
    / k / -1 : h'73657373696f6e6b6579'
  }
  ,
```

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

/ Access Token Response /
Header: Created (Code=2.01)
Content-Format: application/ace+cbor
Max-Age: 3568
Payload:
  
  e'token_upload' : 0,
  
  e'token_hash' : h'0153269057e12fe2b74ba07c892560a2d753877eb62ff44d5a1902530ed97ffe4',
  / expires_in / 2 : 3600,
  / cnf / 8 : {  / COSE_Key / 1 : {  / kty / 1 : 4 / Symmetric /,
    / kid / 2 : h'3d027833fc6267ce',
    / k / -1 : h'73657373696f6e6b6579'
  }
  ,
```

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

/ Access Token Response /
Header: Created (Code=2.01)
Content-Format: application/ace+cbor
Max-Age: 3568
Payload:
  
  e'token_upload' : 0,
  
  e'token_hash' : h'0153269057e12fe2b74ba07c892560a2d753877eb62ff44d5a1902530ed97ffe4',
  / expires_in / 2 : 3600,
  / cnf / 8 : {  / COSE_Key / 1 : {  / kty / 1 : 4 / Symmetric /,
    / kid / 2 : h'3d027833fc6267ce',
    / k / -1 : h'73657373696f6e6b6579'
  }
  ,
```

As it was requested in

As it was requested in
Updates in v -02

Bidirectional access control – The original way
1. DEV1 (C) to DEV2 (RS), with an access token T1 requested by DEV1
2. DEV2 (C) to DEV1 (RS), with another access token T2 requested by DEV2

For the Direction #1, T1 specifies:
- “aud” – An identifier of DEV2
- “scope” – The access rights of DEV1 for accessing DEV2

For the Direction #2, T2 specifies:
- “aud” – An identifier of DEV1
- “scope” – The access rights of DEV2 for accessing DEV1

New proposal: enforce both directions, but with a single access token
- See also the Github issue #1

- DEV1 is registered as:
  - Device authorized to access DEV2; and
  - Device that can be accessed by DEV2
- DEV2 is registered as:
  - Device that can be accessed by DEV1; and
  - Device authorized to access DEV1

Direction #1 (access token T1)
- aud = DEV2
- scope = DEV1@DEV2

Direction #2 (access token T2)
- aud = DEV1
- scope = DEV2@DEV1
Updates in v-02

Bidirectional access control – The new way
1. DEV1 (C) to DEV2 (RS), with an access token T requested by DEV1
2. DEV2 (RS) to DEV1 (C), with the same access token T

Who has which roles?
- DEV1 is the only ACE Client, as the only one requesting an access token
- DEV2 is the only ACE RS, as the only one processing an access token
- DEV1 and DEV2 are CoAP Client and CoAP Server (like for the old way)

For the primary Direction #1, T specifies:
- “aud” – An identifier of DEV2
- “scope” – The access rights of DEV1 for accessing DEV2

For the secondary Direction #2, T specifies:
- “rev_aud” – An identifier of DEV1
- “rev_scope” – The access rights of DEV2 for accessing DEV1

Direction #1 (access token T)
- aud = DEV2
- scope = DEV1@DEV2

Direction #2 (access token T)
- rev_aud = DEV1
- rev_scope = DEV2@DEV1
Updates in v -02

- Defined two new parameters for the Access Token Request/Response
  - “rev_aud”: identifier of the Client C requesting the access token
  - “rev_scope”: access rights that C wishes to grant to the audience
  - While used for the secondary Direction #2, they have the same
    encoding and purpose of “aud” and “scope” for Direction #1
  - Corresponding claims for the access token are also defined

- Practical use (in short)
  - The primary Direction #1 works as usual, based on “aud” and “scope”
  - The Access Token Request can include “rev_aud” and “rev_scope”, i.e.:
    - C asks that the access token also enables the secondary Direction #2
  - The AS determines the granted reverse scope like it does for an Access Token Request where
    the values of “rev_aud” and “rev_scope” are instead specified as values of “aud” and “scope”
Secure communication
- The same secure communication association is used for both directions
- The association is still established per the used transport profile of ACE

Dynamic update of access rights
- Still possible according to the used transport profile of ACE
- Only the access token requester (DEV1) can trigger it

Alternative ACE workflow
- Entirely orthogonal to the bidirectional access control
- Also in this case, the AS can seamlessly upload the access token to DEV2 on behalf of DEV1
Next steps

› Address a comment from IANA on Section 8.2 “OAuth Parameters CBOR Mappings Registry”:
  – Please add an entry for the OAuth Parameters CBOR Mappings field called "Original Specification."

› OAuth parameter “anchor_cnf”
  – Guidelines for its use with group-audiences (see also the Github issue #2)

› Alternative ACE workflow
  – Handling of an uploaded access token on the RS side
  – Ensure applicability to any ACE profile (*)
  – Ensure the possible dynamic update of access rights (*)
  – Detailed use in different transport profiles of ACE

› Bidirectional access control with a single access token (see also the Github issue #1)
  – Setup with two different Authorization Servers
  – More practical considerations on applicability, limitations, and group-audiences

› Comments are welcome!

(*) Appendix B provides a roadmap for addressing this point
Thank you!

Comments/questions?

https://github.com/ace-wg/ace-workflow-and-params
Backup
### Alternative workflow

- **(A) C-to-AS Token Request as usual**
  - C explicitly opts-in for the new workflow, by including the new parameter “token_upload”
  - The final choice about using it is on the AS

- **(A1) The AS uploads the access token to RS, on behalf of C**
  - No intention to replace the original workflow
  - The AS can dynamically choose the workflow to use, e.g., based on the specific RS

- **(A2) The AS receives a response from RS**

- **(B) AS-to-C Token Response**
  - New parameter “token_upload”, with value 0 (successful upload) or 1 (failed upload)
  - 0 → The Response includes: the access token; or a token hash; or neither. Then, C skips step C1.
  - 1 → The Response includes the access token. Then, C performs step C1.
Examples with alternative workflow

Example 1: the AS successfully uploaded the access token

```
Header: Created (Code=2.01)
Content-Format: application/ace+cbor
Max-Age: 3560
Payload:
{
  "token_upload": 0,
  / "expires_in": 2:3600,
  / "cnf": 8:
  / "COSF_Key": 1:
    / "kty": 1:4:Symmetric /
    / "kid": 2: 1:3d0278334c5267ce /
    / "k": 1:73657373696f6e6e579
}
```

Example 2: the AS attempted to upload the access token but failed

```
Header: Created (Code=2.01)
Content-Format: application/ace+cbor
Max-Age: 3560
Payload:
{
  "token_upload": 1,
  / "access_token": 1: h'd08343a1'/ ...
  (remainder of CWT omitted for brevity;
  CWT contains the symmetric PoP key in the "cnf" claim)/,
  / "expires_in": 2:3600,
  / "cnf": 8:
  / "COSF_Key": 1:
    / "kty": 1:4:Symmetric /
    / "kid": 2: 1:3d0278334c5267ce /
    / "k": 1:73657373696f6e6e579
}
```