ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 16 September 2022

C. Bormann
Universität Bremen TZI

15 March 2022

An Authorization Information Format (AIF) for ACE
draft-ietf-ace-aif-07

Abstract

Information about which entities are authorized to perform what operations on which constituents of other entities is a crucial component of producing an overall system that is secure. Conveying precise authorization information is especially critical in highly automated systems with large numbers of entities, such as the "Internet of Things".

This specification provides a generic information model and format for representing such authorization information, as well as two variants of a specific instantiation of that format for use with REST resources identified by URI path.

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-aif/.

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/.

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 16 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 3
1.1. Terminology 3
2. Information Model 3
2.1. REST-specific Model 4
2.2. Limitations 5
2.3. REST-specific Model With Dynamic Resource Creation . . . 6
3. Data Model 6
4. Media Types 9
5. IANA Considerations 9
5.1. Media Types 9
5.2. Registries 11
5.3. Content-Format 12
6. Security Considerations 12
7. References 13
7.1. Normative References 13
7.2. Informative References 14
Acknowledgements 16
Author's Address 16

1. Introduction

Constrained Devices as they are used in the "Internet of Things" need security in order to operate correctly and prevent misuse. One important element of this security is that devices in the Internet of Things need to be able to decide which operations requested of them should be considered authorized, need to ascertain that the authorization to request the operation does apply to the actual requester as authenticated, and need to ascertain that other devices they make requests of are the ones they intended.

To transfer detailed authorization information from an authorization manager (such as an ACE-OAuth Authorization Server [I-D.ietf-ace-oauth-authz]) to a device, a compact representation format is needed. This document defines such a format, the Authorization Information Format (AIF). AIF is defined both as a general structure that can be used for many different applications and as a specific instantiation tailored to REST resources and the permissions on them, including some provision for dynamically created resources.

1.1. Terminology

This memo uses terms from CoAP [RFC7252] and the Internet Security Glossary [RFC4949]; CoAP is used for the explanatory examples as it is a good fit for Constrained Devices.

The shape of data is specified in CDDL [RFC8610] [RFC9165]. Terminology for Constrained Devices is defined in [RFC7228].

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.

The term "byte", abbreviated by "B", is used in its now customary sense as a synonym for "octet".

2. Information Model

Authorizations are generally expressed through some data structures that are cryptographically secured (or transmitted in a secure way). This section discusses the information model underlying the payload of that data (as opposed to the cryptographic armor around it).

The semantics of the authorization information defined in this document are that of an _allow-list_: everything is denied until it is explicitly allowed.

For the purposes of this specification, the underlying access control model will be that of an access matrix, which gives a set of permissions for each possible combination of a subject and an object. We are focusing the AIF data item on a single row in the access matrix (such a row has often been called a capability list), without concern to the subject for which the data item is issued. As a consequence, AIF MUST be used in a way that the subject of the authorizations is unambiguously identified (e.g., as part of the armor around it).

The generic model of such a capability list is a list of pairs of object identifiers (of type Toid) and the permissions (of type Tperm) the subject has on the object(s) identified.

```
AIF-Generic<Toid, Tperm> = [* [Toid, Tperm]]
```

Figure 1: Definition of Generic AIF

In a specific data model (such as the one also specified in this document), the object identifier (Toid) will often be a text string, and the set of permissions (Tperm) will be represented by a bitset in turn represented as a number (see Section 3).

```
AIF-Specific = AIF-Generic<tstr, uint>
```

Figure 2: Commonly used shape of a specific AIF

2.1. REST-specific Model

In the specific instantiation of the REST resources and the permissions on them, for the object identifiers (Toid), we use the URI of a resource on a CoAP server. More specifically, since the parts of the URI that identify the server ("authority" in [RFC3986]) are what are authenticated during REST resource access (Section 4.2.2 of [I-D.ietf-httpbis-semantics] and Section 6.2 of [RFC7252]), they naturally fall into the realm handled by the cryptographic armor; we therefore focus on the "path" ("path-abempty") and "query" parts of the URI (_URI-local-part_ in this specification, as expressed by the Uri-Path and Uri-Query options in CoAP). As a consequence, AIF MUST be used in a way that it is clear who is the target (enforcement point) of these authorizations (note that there may be more than one target that the same authorization applies to, e.g., in a situation with homogeneous devices).

For the permissions (Tperm), we use a simple permissions model that lists the subset of the REST (CoAP or HTTP) methods permitted. This model is summarized in Table 1.

URI-local-part	Permission Set
/s/temp	GET
/a/led	PUT, GET
/dtls	POST

Table 1: An authorization instance in the AIF Information Model

In this example, a device offers a temperature sensor /s/temp for read-only access, a LED actuator /a/led for read/write, and a /dtls resource for POST access.

As will be seen in the data model (Section 3), the representations of REST methods provided are limited to those that have a CoAP method number assigned; an extension to the model may be necessary to represent permissions for exotic HTTP methods.

2.2. Limitations

This simple information model only allows granting permissions for statically identifiable objects, e.g., URIs for the REST-specific instantiation. One might be tempted to extend the model towards URI templates [RFC6570] (for instance, to open up an authorization for many parameter values as in /s/temp{?any*}). However, that requires some considerations of the ease and unambiguity of matching a given URI against a set of templates in an AIF data item.

This simple information model also does not allow expressing conditionalized access based on state outside the identification of objects (e.g., "opening a door is allowed if that is not locked").

Finally, the model does not provide any special access for a set of resources that are specific to a subject, e.g., that the subject created itself by previous operations (PUT, POST, or PATCH/iPATCH [RFC8132]) or that were specifically created for the subject by others.

2.3. REST-specific Model With Dynamic Resource Creation

The REST-specific Model With Dynamic Resource Creation addresses the need to provide defined access to dynamic resources that were created by the subject itself, specifically, a resource that is made known to the subject by providing Location-* options in a CoAP response or using the Location header field in HTTP [I-D.ietf-httpbis-semantics] (the Location-indicating mechanisms). (The concept is somewhat comparable to "ACL inheritance" in NFSv4 [RFC8881], except that it does not use a containment relationship but the fact that the dynamic resource was created from a resource to which the subject had access.) In other words, it addresses an important subset of the third limitation mentioned in Section 2.2.

URI-local-part	Permission Set
/a/make-coffee	POST, Dynamic-GET, Dynamic-DELETE

Table 2: An authorization instance in the AIF Information Model With Dynamic Resource Creation

For a method X, the presence of a Dynamic-X permission means that the subject holds permission to exercise the method X on resources that have been returned in a 2.01 (201 Created) response by a Location-indicating mechanism to a request that the subject made to the resource listed. In the example shown in Table 2, POST operations on /a/make-coffee might return the location of a resource dynamically created on the coffee machine that allows GET to find out about the status of, and DELETE to cancel, the coffee-making operation.

Since the use of the extension defined in this section can be detected by the mentioning of the Dynamic-X permissions, there is no need for another explicit switch between the basic and the model extended by dynamic resource creation; the extended model is always presumed once a Dynamic-X permission is present.

3. Data Model

Different data model specializations can be defined for the generic information model given above.

In this section, we will give the data model for simple REST authorization as per Section 2.1 and Section 2.3. As discussed, in this case the object identifier is specialized as a text string giving a relative URI (URI-local-part as absolute path on the server serving as enforcement point). The permission set is specialized to a single number REST-method-set by the following steps:

* The entries in the table that specify the same URI-local-part are merged into a single entry that specifies the union of the permission sets.

* The (non-dynamic) methods in the permission sets are converted into their CoAP method numbers, minus 1.

* Dynamic-X permissions are converted into what the number would have been for X, plus a Dynamic-Offset chosen as 32 (e.g., 35 is the number for Dynamic-DELETE as the number for DELETE is 3).

* The set of numbers is converted into a single number REST-method-set by taking two to the power of each (decremented) method number and computing the inclusive OR of the binary representations of all the power values.

This data model could be interchanged in the JSON [RFC8259] representation given in Figure 3.

```
[["/s/temp",1],["/a/led",5],["/dtls",2]]
```

Figure 3: An authorization instance encoded in JSON (40 bytes)

In Figure 4, a straightforward specification of the data model (including both the methods from [RFC7252] and the new ones from [RFC8132], identified by the method code minus 1) is shown in CDDL [RFC8610] [RFC9165]:

```
AIF-REST = AIF-Generic<local-path, REST-method-set>
local-path = tstr   ; URI relative to enforcement point
REST-method-set = uint .bits methods
methods = &(
  GET: 0
  POST: 1
  PUT: 2
  DELETE: 3
  FETCH: 4
  PATCH: 5
  iPATCH: 6
  Dynamic-GET: 32; 0 .plus Dynamic-Offset
  Dynamic-POST: 33; 1 .plus Dynamic-Offset
  Dynamic-PUT: 34; 2 .plus Dynamic-Offset
  Dynamic-DELETE: 35; 3 .plus Dynamic-Offset
  Dynamic-FETCH: 36; 4 .plus Dynamic-Offset
  Dynamic-PATCH: 37; 5 .plus Dynamic-Offset
  Dynamic-iPATCH: 38; 6 .plus Dynamic-Offset
)
```

Figure 4: AIF in CDDL

For the information shown in Table 1 and Figure 3, a representation in CBOR [RFC8949] is given in Figure 5; again, several optimizations/ improvements are possible.

```
83                      # array(3)
   82                   # array(2)
      67                # text(7)
         2f732f74656d70  # "/s/temp"
      01                # unsigned(1)
   82                   # array(2)
      66                # text(6)
         2f612f6c6564    # "/a/led"
      05                # unsigned(5)
   82                   # array(2)
      65                # text(5)
         2f64746c73      # "/dtls"
      02                # unsigned(2)
```

Figure 5: An authorization instance encoded in CBOR (28 bytes)

Note that choosing 32 as Dynamic-Offset means that all future CoAP methods that can be registered can be represented both as themselves and in the Dynamic-X variant, but only the dynamic forms of methods 1 to 21 are typically usable in a JSON form [RFC7493].

4. Media Types

This specification defines media types for the generic information model, expressed in JSON (application/aif+json) or in CBOR (application/aif+cbor). These media types have parameters for specifying Toid and Tperm; default values are the values "URI-local-part" for Toid and "REST-method-set" for Tperm, as per Section 3 of the present specification.

A specification that wants to use Generic AIF with different Toid and/or Tperm is expected to request these as media type parameters (Section 5.2) and register a corresponding Content-Format (Section 5.3).

5. IANA Considerations

// RFC Ed.: throughout this section, please replace RFC XXXX with the
// RFC number of this specification and remove this note.

5.1. Media Types

IANA is requested to add the following Media-Types to the "Media Types" registry.

Name	Template	Reference
aif+cbor	application/aif+cbor	RFC XXXX, Section 4
aif+json	application/aif+json	RFC XXXX, Section 4

Table 3: New Media Types

For application/aif+cbor:

Type name: application
Subtype name: aif+cbor
Required parameters: N/A
Optional parameters:

* Toid: the identifier for the object for which permissions are supplied. A value from the media-type parameter sub-registry for Toid. Default value: "URI-local-part" (RFC XXXX).

* Tperm: the data type of a permission set for the object identified via a Toid. A value from the media-type parameter sub-registry for Tperm. Default value: "REST-method-set" (RFC XXXX).

Encoding considerations: binary (CBOR)
Security considerations: Section 6 of RFC XXXX
Interoperability considerations: none
Published specification: Section 4 of RFC XXXX
Applications that use this media type: Applications that need to convey structured authorization data for identified resources, conveying sets of permissions.
Fragment identifier considerations: The syntax and semantics of fragment identifiers is as specified for "application/cbor". (At publication of RFC XXXX, there is no fragment identification syntax defined for "application/cbor".)
Person & email address to contact for further information: ACE WG mailing list (ace@ietf.org), or IETF Applications and Real-Time Area (art@ietf.org)
Intended usage: COMMON
Restrictions on usage: none
Author/Change controller: IETF
Provisional registration: no

For application/aif+json:

Type name: application
Subtype name: aif+json
Required parameters: N/A
Optional parameters:

* Toid: the identifier for the object for which permissions are supplied. A value from the media-type parameter sub-registry for Toid. Default value: "URI-local-part" (RFC XXXX).

* Tperm: the data type of a permission set for the object identified via a Toid. A value from the media-type parameter sub-registry for Tperm. Default value: "REST-method-set" (RFC XXXX).

Encoding considerations: binary (JSON is UTF-8-encoded text)
Security considerations: Section 6 of RFC XXXX
Interoperability considerations: none
Published specification: Section 4 of RFC XXXX
Applications that use this media type: Applications that need to convey structured authorization data for identified resources, conveying sets of permissions.
Fragment identifier considerations: The syntax and semantics of fragment identifiers is as specified for "application/json". (At publication of RFC XXXX, there is no fragment identification syntax defined for "application/json".)

Person & email address to contact for further information: ACE WG mailing list (ace@ietf.org), or IETF Applications and Real-Time Area (art@ietf.org)
Intended usage: COMMON
Restrictions on usage: none
Author/Change controller: IETF
Provisional registration: no

5.2. Registries

For the media types application/aif+cbor and application/aif+json, IANA is requested to create a sub-registry within [IANA.media-type-sub-parameters] for the two media-type parameters Toid and Tperm, populated with:

Parameter	name	Description/ Specification	Reference
Toid	URI-local-part	local-part of URI	RFC XXXX
Tperm	REST-method-set	set of REST methods represented	RFC XXXX

Table 4: New Media Type Parameters

The registration policy is Specification required [RFC8126]. The designated expert will engage with the submitter to ascertain the requirements of this document are addressed:

* The specifications for Toid and Tperm need to realize the general ideas of unambiguous object identifiers and permission lists in the context where the AIF data item is intended to be used, and their structure needs to be usable with the intended media types (application/aif+cbor and application/aif+json) as identified in the specification.

* The parameter names need to conform to Section 4.3 of [RFC6838], but preferably are in [KebabCase] so they can easily be translated into names used in popular programming language APIs.

The designated experts will develop further criteria and guidelines as needed.

5.3. Content-Format

IANA is requested to register Content-Format numbers in the "CoAP Content-Formats" sub-registry, within the "Constrained RESTful Environments (CoRE) Parameters" Registry [IANA.core-parameters], as follows:

Content-Type	Content Coding	ID	Reference
application/aif+cbor	-	TBD1	RFC XXXX
application/aif+json	-	TBD2	RFC XXXX

Table 5: New Content-Formats

// RFC Ed.: please replace TBD1 and TBD2 with assigned IDs and remove this note.

In the registry as defined by Section 12.3 of [RFC7252] at the time of writing, the column "Content-Type" is called "Media type" and the column "Content Coding" is called "Encoding".

Note that applications that register Toid and Tperm values are encouraged to also register Content-Formats for the relevant combinations.

6. Security Considerations

The security considerations of [RFC7252] apply when AIF is used with CoAP, and, if complex formats such as URIs are used for Toid or Tperm, specifically Section 11.1 of [RFC7252]. Some wider issues are discussed in [RFC8576].

When applying these formats, the referencing specification needs to be careful to:

* ensure that the cryptographic armor employed around this format fulfills the referencing specification’s security objectives, and that the armor or some additional information included in it with the AIF data item (1) unambiguously identifies the subject to which the authorizations shall apply and (2) provides any context information needed to derive the identity of the object to which authorization is being granted from the object identifiers (such as, for the data models defined in the present specification, the scheme and authority information that is used to construct the full URI), and

* ensure that the types used for Toid and Tperm provide the appropriate granularity and precision so that application requirements on the precision of the authorization information are fulfilled, and that all parties have the same understanding of each Toid/Tperm pair in terms of specified objects (resources) and operations on those.

For the data formats, the security considerations of [RFC8259] and [RFC8949] apply.

A plain implementation of AIF might implement just the basic REST model as per Section 2.1. If it receives authorizations that include permissions that use the REST-specific Model With Dynamic Resource Creation Section 2.3, it needs to either reject the AIF data item entirely or act only on the permissions that it does understand. In other words, the semantics underlying an allow-list as discussed above need to hold here as well.

An implementation of the REST-specific Model With Dynamic Resource Creation Section 2.3 needs to carefully keep track of the dynamically created objects and the subjects to which the Dynamic-X permissions apply -- both on the server side to enforce the permissions and on the client side to know which permissions are available.

7. References

7.1. Normative References

[I-D.ietf-httpbis-semantics]
Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Semantics", Work in Progress, Internet-Draft, draft-ietf-httpbis-semantics-19, 12 September 2021, <https://www.ietf.org/archive/id/draft-ietf-httpbis-semantics-19.txt>.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

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

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

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

[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/info/rfc8610>.

[RFC9165] Bormann, C., "Additional Control Operators for the Concise Data Definition Language (CDDL)", RFC 9165, DOI 10.17487/RFC9165, December 2021, <https://www.rfc-editor.org/info/rfc9165>.

7.2. Informative References

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", Work in Progress, Internet-Draft, draft-ietf-ace-oauth-authz-46, 8 November 2021, <https://www.ietf.org/archive/id/draft-ietf-ace-oauth-authz-46.txt>.

[IANA.core-parameters]
IANA, "Constrained RESTful Environments (CoRE) Parameters", <https://www.iana.org/assignments/core-parameters>.

[IANA.media-type-sub-parameters]
IANA, "MIME Media Type Sub-Parameter Registries", <https://www.iana.org/assignments/media-type-sub-parameters>.

[KebabCase]
"KebabCase", 29 August 2014, <http://wiki.c2.com/?KebabCase>.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2", FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007, <https://www.rfc-editor.org/info/rfc4949>.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <https://www.rfc-editor.org/info/rfc6570>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/info/rfc7228>.

[RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <https://www.rfc-editor.org/info/rfc7493>.

[RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017, <https://www.rfc-editor.org/info/rfc8132>.

[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/info/rfc8259>.

[RFC8576] Garcia-Morchon, O., Kumar, S., and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges", RFC 8576, DOI 10.17487/RFC8576, April 2019, <https://www.rfc-editor.org/info/rfc8576>.

[RFC8881] Noveck, D., Ed. and C. Lever, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 8881, DOI 10.17487/RFC8881, August 2020, <https://www.rfc-editor.org/info/rfc8881>.

[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/info/rfc8949>.

Acknowledgements

Jim Schaad, Francesca Palombini, Olaf Bergmann, Marco Tiloca, and Christian Amsüss provided comments that shaped the direction of this document. Alexey Melnikov pointed out that there were gaps in the media type specifications, and Loganaden Velvindron provided a shepherd review with further comments. Many thanks also to the IESG reviewers, which provided several small but significant observations. Benjamin Kaduk provided an extensive review as responsible Area Director, and indeed is responsible for much improvement in the document.

Author's Address

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org

ACE
Internet-Draft
Intended status: Standards Track
Expires: 16 November 2023

M. Sahni, Ed.
S. Tripathi, Ed.
Palo Alto Networks

15 May 2023

CoAP Transfer for the Certificate Management Protocol
draft-ietf-ace-cmpv2-coap-transport-10

Abstract

This document specifies the use of Constrained Application Protocol (CoAP) as a transfer mechanism for the Certificate Management Protocol (CMP). CMP defines the interaction between various PKI entities for the purpose of certificate creation and management. CoAP is an HTTP-like client-server protocol used by various constrained devices in the IoT space.

Status of This Memo

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

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

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

This Internet-Draft will expire on 16 November 2023.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 2
1.1. Terminology 3
2. CoAP Transfer Mechanism for CMP 3
2.1. CoAP URI Format 3
2.2. Discovery of CMP RA/CA 4
2.3. CoAP Request Format 4
2.4. CoAP Block-Wise Transfer Mode 4
2.5. Multicast CoAP 5
2.6. Announcement PKIMessage 5
3. Proxy Support 6
4. Security Considerations 6
5. IANA Considerations 7
6. Acknowledgments 8
7. References 8
7.1. Normative References 8
7.2. Informative References 10
Authors' Addresses 10

1. Introduction

The Certificate Management Protocol (CMP) [RFC4210] is used by the PKI entities for the generation and management of certificates. One of the requirements of Certificate Management Protocol is to be independent of the transport protocol in use. CMP has mechanisms to take care of required transactions, error reporting and protection of messages.

The Constrained Application Protocol (CoAP) defined in [RFC7252], [RFC7959] and [RFC8323] is a client-server protocol like HTTP. It is designed to be used by constrained devices over constrained networks. The recommended transport for CoAP is UDP, however [RFC8323] specifies the support of CoAP over TCP, TLS and Websockets.

This document specifies the use of CoAP over UDP as a transport medium for the CMP version 2 [RFC4210], CMP version 3 [I-D.ietf-lamps-cmp-updates] designated as CMP in this document and Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile]. This document, in general, follows the HTTP transfer for CMP specifications defined in [RFC6712] and specifies the requirements for using CoAP as a transfer mechanism for the CMP.

This document also provides guidance on how to use a "CoAP-to-HTTP" proxy to ease adoption of CoAP transfer mechanism by enabling the interconnection with existing PKI entities already providing CMP over HTTP.

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.

2. CoAP Transfer Mechanism for CMP

A CMP transaction consists of exchanging PKIMessages [RFC4210] between PKI End Entities (EEs), Registration Authorities (RAs), and Certification Authorities (CAs). If the EEs are constrained devices then they may prefer, as a CMP client, the use of CoAP instead of HTTP as the transfer mechanism. The RAs and CAs, in general, are not constrained and can support both CoAP and HTTP Client and Server implementations. This section specifies how to use CoAP as the transfer mechanism for the Certificate Management Protocol.

2.1. CoAP URI Format

The CoAP URI format is described in section 6 of [RFC7252]. The CoAP endpoints MUST support use of the path prefix "/.well-known/" as defined in [RFC8615] and the registered name "cmp" to help with endpoint discovery and interoperability. Optional path segments MAY be added after the registered application name (i.e. after "/.well-known/cmp") to provide distinction. The path segment 'p' followed by an arbitraryLabel <name> could for example support the differentiation of specific CAs or certificate profiles. Further path segments, e.g., as specified in the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile], could indicate PKI management operations using an operationLabel <operation>. A valid full CMP URI can look like this:

```
coap://www.example.com/.well-known/cmp
coap://www.example.com/.well-known/cmp/<operation>
coap://www.example.com/.well-known/cmp/p/<profileLabel>
coap://www.example.com/.well-known/cmp/p/<profileLabel>/<operation>
```

2.2. Discovery of CMP RA/CA

The EEs can be configured with enough information to form the CMP server URI. The minimum information that can be configured is the scheme i.e. "coap:" or "coaps:" and the authority portion of the URI, e.g. "example.com:5683". If the port number is not specified in the authority, then the default ports numbers MUST be assumed for the "coap:" and the "coaps:" scheme URIs. The default port for coap: scheme URIs is 5683 and the default port for coaps: scheme URIs is 5684 [RFC7252].

Optionally, in the environments where a Local Registration Authority (LRA) or a Local CA is deployed, EEs can also use the CoAP service discovery mechanism [RFC7252] to discover the URI of the Local RA or CA. The CoAP CMP endpoints supporting service discovery MUST also support resource discovery in the CoRE Link Format as described in [RFC6690]. The Link MUST include the 'ct' attribute defined in section 7.2.1 of [RFC7252] with the value of "application/pkixcmp" as defined in the CoAP Content-Formats IANA registry.

2.3. CoAP Request Format

The CMP PKIMessages MUST be DER encoded and sent as the body of the CoAP POST request. A CMP client MUST send each CoAP requests marked as a Confirmable message [RFC7252]. If the CoAP request is successful then the CMP RA or CA MUST return a Success 2.xx response code otherwise CMP RA or CA MUST return an appropriate Client Error 4.xx or Server Error 5.xx response code. A CMP RA or CA may choose to send a Piggybacked response [RFC7252] to the client or it MAY send a Separate response [RFC7252] in case it takes some time for CA or RA to process the CMP transaction.

When transferring CMP PKIMesssage over CoAP the content-format "application/pkixcmp" MUST be used.

2.4. CoAP Block-Wise Transfer Mode

A CMP PKIMesssage consists of a header, body, protection, and extraCerts structures which may contain many optional and potentially large fields. Thus, a CMP message can be much larger than the Maximum Transmission Unit (MTU) of the outgoing interface of the device. The EEs and RAs or CAs, MUST use the Block-Wise transfer mode [RFC7959] to transfer such large messages instead of relying on IP fragmentation.

If a CoAP-to-HTTP proxy is in the path between EEs and CA or EEs and RA then, if the server supports, it MUST use the chunked transfer encoding [RFC9112] to send data over the HTTP transport. The proxy MUST try to reduce the number of packets sent by using an optimal chunk length for the HTTP transport.

2.5. Multicast CoAP

CMP PKIMessages sent over CoAP MUST NOT use a Multicast destination address.

2.6. Announcement PKIMessage

A CMP server may publish announcements, that can be event triggered or periodic, for the other PKI entities. Here is the list of CMP announcement messages prefixed by their respective ASN.1 identifier (section 5.1.2 [RFC4210])

[15] CA Key Update Announcement
[16] Certificate Announcement
[17] Revocation Announcement
[18] CRL Announcement

An EE MAY use CoAP Observe option [RFC7641] to register itself to get any announcement messages from the RA or CA. The EE can send a GET request to the server's URI suffixed by "/ann". For example a path to register for announcement messages may look like this:

coap://www.example.com/.well-known/cmp/ann
coap://www.example.com/.well-known/cmp/p/<profileLabel>/ann

If the server supports CMP Announcements messages, then it MUST send appropriate Success 2.xx response code, otherwise it MUST send an appropriate Client Error 4.xx or Server Error 5.xx response code. If for some reason the server cannot add the client to its list of observers for the announcements, it can omit the Observe option [RFC7641] in the response to the client. A client on receiving a 2.xx success response without the Observe option [RFC7641] MAY try after some time to register again for announcements from the CMP server. Since server can remove the EE from the list of observers for announcement messages, an EE SHOULD periodically re-register itself for announcement messages.

Alternatively, an EE MAY periodically poll for the current status of the CA via the "PKI Information Request" message, see section 6.5 of [RFC4210]. If supported, EEs MAY also use "Support Messages" defined in section 4.3 of Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile] to get information about the CA status. These mechanisms will help constrained devices, that are acting as EEs, to conserve resources by eliminating the need to create an endpoint for receiving notifications from RA or CA. It will also simplify the implementation of a CoAP-to-HTTP proxy.

3. Proxy Support

This section provides guidance on using a CoAP-to-HTTP proxy between EEs and RAs or CAs in order to avoid changes to the existing PKI implementation.

Since CMP payload is the same over CoAP and HTTP transfer mechanisms, a CoAP-to-HTTP cross-protocol proxy can be implemented based on section 10 of [RFC7252]. The CoAP-to-HTTP proxy can either be located closer to the EEs or closer to the RA or CA. The proxy MAY support service discovery and resource discovery as described in section 2.2. The CoAP-to-HTTP proxy MUST function as a reverse proxy, only permitting connections to a limited set of pre-configured servers. It is out of scope of this document to specify how a reverse proxy can route CoAP client requests to one of the configured servers. Some recommended mechanisms are as follows:

* Use the Uri-Path option to identify a server.

* Use separate hostnames for each of the configured servers and then use the Uri-Host option for routing the CoAP requests.

* Use separate hostnames for each of the configured servers and then use Server Name Indication [RFC8446] in case of "coaps://" scheme for routing CoAP requests.

4. Security Considerations

* If PKIProtection is used, the PKIHeader and PKIBody of the CMP protocol are cryptographically protected against malicious modifications. As such, UDP can be used without compromising the security of the CMP protocol. Security Considerations for CoAP are defined in [RFC7252].

* The CMP protocol does not provide confidentiality of the CMP payloads. If confidentiality is desired, CoAP over DTLS [RFC9147] SHOULD be used to provide confidentiality for the CMP payloads, although it cannot conceal that the CMP protocol is used within the DTLS layer.
* Section 9.1 of [RFC7252] defines how to use DTLS [RFC9147] for securing the CoAP. DTLS [RFC9147] associations SHOULD be kept alive and re-used where possible to amortize on the additional overhead of DTLS on constrained devices.
* An EE might not witness all of the Announcement messages when using the CoAP Observe option [RFC7641], since the Observe option is a "best-effort" approach and the server might lose its state for subscribers to its announcement messages. The EEs may use an alternate method described in section 2.6 to obtain time critical changes such as CRL [RFC5280] updates.
* Implementations SHOULD use the available datagram size and avoid sending small datagrams containing partial CMP PKIMessage data in order to reduce memory usage for packet buffering.
* A CoAP-to-HTTP proxy can also protect the PKI entities by handling UDP and CoAP messages. The proxy can mitigate attacks like denial of service attacks, replay attacks and resource-exhaustion attacks by enforcing basic checks like validating that the ASN.1 syntax is compliant to CMP messages and validating the PKIMessage protection before sending them to PKI entities.
* Since the Proxy may have access to the CMP-Level metadata and control over the flow of CMP messages therefore proper role based access control should be in place. The proxy can be deployed at the edge of the "End Entities" network or in front of an RA and CA to protect them. The proxy however may itself be vulnerable to resource-exhaustion attacks as it’s required to buffer the CMP messages received over CoAP transport before sending it to the HTTP endpoint. This can be mitigated by using short timers for discarding the buffered messages and rate limiting clients based on the resource usage.

5. IANA Considerations

This document adds a new entry to the CoAP Content-Formats IANA Registry (https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats) for the code of content-type "application/pkixcmp", for transferring CMP transactions over CoAP, from the identifier range 256-9999 reserved for IETF specifications.

Type name: application

Subtype name: pkixcmp

Encoding: Content may contain arbitrary octet values. The octet values are the ASN.1 DER encoding of a PKI message, as defined in the [RFC4210] specifications.

Reference: This document and [RFC4210]

This document also adds a new path segment "ann" to the CMP Well-Known URI Path Segments (https://www.iana.org/assignments/cmp/cmp.xhtml#cmp-well-known-uri) IANA registry for the EEs to register themselves for the announcement messages.

Path Segment: ann

Description: The path to send a GET request with CoAP Observer Option to register for CMP announcement messages.

Reference: This document.

This document references the cmp, in the Well-Known URIs (https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml) IANA registry. Please add a reference of this document to the Well-Known URIs (https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml) IANA registry for that entry.

This document also refers the path segment "p" in the CMP Well-Known URI Path Segments (https://www.iana.org/assignments/cmp/cmp.xhtml#cmp-well-known-uri) IANA registry. Please add a reference of this document to the CMP Well-Known URI Path Segments (https://www.iana.org/assignments/cmp/cmp.xhtml#cmp-well-known-uri) for that path segment.

[Note RFC Editor]: This document should be published together or after the CMP version 3 [I-D.ietf-lamps-cmp-updates] as it references IANA entries created by that Internet draft.

6. Acknowledgments

The authors would like to thank Hendrik Brockhaus, David von Oheimb, and Andreas Kretschmer for their guidance in writing the content of this document and providing valuable feedback.

7. References

7.1. Normative References

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

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

[RFC6712] Kause, T. and M. Peylo, "Internet X.509 Public Key Infrastructure -- HTTP Transfer for the Certificate Management Protocol (CMP)", RFC 6712, DOI 10.17487/RFC6712, September 2012, <https://www.rfc-editor.org/info/rfc6712>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

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

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[I-D.ietf-lamps-cmp-updates] Brockhaus, H., von Oheimb, D., and J. Gray, "Certificate Management Protocol (CMP) Updates", Work in Progress, Internet-Draft, draft-ietf-lamps-cmp-updates-23, 29 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-cmp-updates-23>.

[I-D.ietf-lamps-lightweight-cmp-profile] Brockhaus, H., von Oheimb, D., and S. Fries, "Lightweight Certificate Management Protocol (CMP) Profile", Work in Progress, Internet-Draft, draft-ietf-lamps-lightweight-cmp-profile-21, 17 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-lightweight-cmp-profile-21>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/info/rfc6690>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

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

[RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/info/rfc9112>.

7.2. Informative References

[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/info/rfc5280>.

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

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/info/rfc8323>.

Authors' Addresses

Mohit Sahni (editor)
Palo Alto Networks
3000 Tannery Way
Santa Clara, CA 95054
United States of America
Email: msahni@paloaltonetworks.com

Saurabh Tripathi (editor)
Palo Alto Networks
3000 Tannery Way
Santa Clara, CA 95054
United States of America
Email: stripathi@paloaltonetworks.com

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 19 July 2024

F. Palombini
Ericsson AB
M. Tiloca
RISE AB

16 January 2024

Key Provisioning for Group Communication using ACE draft-ietf-ace-key-groupcomm-18

Abstract

This document defines how to use the Authentication and Authorization for Constrained Environments (ACE) framework to distribute keying material and configuration parameters for secure group communication. Candidate group members acting as Clients and authorized to join a group can do so by interacting with a Key Distribution Center (KDC) acting as Resource Server, from which they obtain the keying material to communicate with other group members. While defining general message formats as well as the interface and operations available at the KDC, this document supports different approaches and protocols for secure group communication. Therefore, details are delegated to separate application profiles of this document, as specialized instances that target a particular group communication approach and define how communications in the group are protected. Compliance requirements for such application profiles are also specified.

Discussion Venues

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 19 July 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 4
1.1. Terminology 5
2. Overview 8
3. Authorization to Join a Group 12
3.1. Authorization Request 12
3.2. Authorization Response 15
3.3. Token Transferring 16
3.3.1. 'sign_info' Parameter 18
3.3.2. 'kdcchallenge' Parameter 21
4. KDC Functionalities 21
4.1. Interface at the KDC 21
4.1.1. Operations Supported by Clients 26
4.1.2. Error Handling 27
4.2. /ace-group 29
4.2.1. FETCH Handler 29
4.2.1.1. Retrieve Group Names 30
4.3. /ace-group/GROUPNAME 31
4.3.1. POST Handler 31
4.3.1.1. Join the Group 47
4.3.2. GET Handler 49
4.3.2.1. Retrieve Group Keying Material 50
4.4. /ace-group/GROUPNAME/creds 51
4.4.1. FETCH Handler 51
4.4.1.1. Retrieve a Subset of Authentication Credentials in the Group 54
4.4.2. GET Handler 55
4.4.2.1. Retrieve All Authentication Credentials in the Group 55
4.5. /ace-group/GROUPNAME/kdc-cred 56
4.5.1. GET Handler 56

4.5.1.1. Retrieve the KDC's Authentication Credential . . 58
4.6. /ace-group/GROUPNAME/policies 59
4.6.1. GET Handler 59
4.6.1.1. Retrieve the Group Policies 60
4.7. /ace-group/GROUPNAME/num 60
4.7.1. GET Handler 60
4.7.1.1. Retrieve the Keying Material Version 61
4.8. /ace-group/GROUPNAME/nodes/NODENAME 62
4.8.1. GET Handler 62
4.8.1.1. Retrieve Group and Individual Keying Material . . 63
4.8.2. PUT Handler 65
4.8.2.1. Request to Change Individual Keying Material . . 67
4.8.3. DELETE Handler 68
4.8.3.1. Leave the Group 69
4.9. /ace-group/GROUPNAME/nodes/NODENAME/cred 69
4.9.1. POST Handler 69
4.9.1.1. Uploading an Authentication Credential 72
5. Removal of a Group Member 73
6. Group Rekeying Process 74
6.1. Point-to-Point Group Rekeying 76
6.2. One-to-Many Group Rekeying 78
6.2.1. Protection of Rekeying Messages 82
6.3. Misalignment of Group Keying Material 85
7. Extended Scope Format 85
8. ACE Groupcomm Parameters 87
9. ACE Groupcomm Error Identifiers 91
10. Security Considerations 93
10.1. Secure Communication in the Group 93
10.2. Update of Group Keying Material 94
10.3. Block-Wise Considerations 96
11. IANA Considerations 96
11.1. Media Type Registrations 97
11.2. CoAP Content-Formats 97
11.3. OAuth Parameters 98
11.4. OAuth Parameters CBOR Mappings 98
11.5. Interface Description (if=) Link Target Attribute Values 99
11.6. Custom Problem Detail Keys Registry 99
11.7. ACE Groupcomm Parameters 100
11.8. ACE Groupcomm Key Types 100
11.9. ACE Groupcomm Profiles 101
11.10. ACE Groupcomm Policies 102
11.11. Sequence Number Synchronization Methods 103
11.12. ACE Groupcomm Errors 104
11.13. ACE Groupcomm Rekeying Schemes 105
11.14. Expert Review Instructions 106
12. References 106
12.1. Normative References 106

12.2. Informative References 109
Appendix A. Requirements for Application Profiles 111
A.1. Mandatory-to-Address Requirements 111
A.2. Optional-to-Address Requirements 114
Appendix B. Extensibility for Future COSE Algorithms 115
B.1. Format of 'sign_info_entry' 115
Appendix C. Document Updates 116
C.1. Version -17 to -18 116
C.2. Version -16 to -17 118
C.3. Version -15 to -16 119
C.4. Version -14 to -15 119
C.5. Version -13 to -14 119
C.6. Version -05 to -13 120
C.7. Version -04 to -05 120
C.8. Version -03 to -04 121
C.9. Version -02 to -03 121
C.10. Version -01 to -02 122
C.11. Version -00 to -01 122
Acknowledgments 123
Authors' Addresses 124

1. Introduction

This document builds on the Authentication and Authorization for Constrained Environments (ACE) framework and defines how to request, distribute, and renew keying material and configuration parameters to protect message exchanges in a group communication environment.

Candidate group members acting as ACE Clients and authorized to join a group can interact with the Key Distribution Center (KDC) acting as ACE Resource Server and responsible for that group, in order to obtain the necessary keying material and parameters to communicate with other group members.

In particular, this document defines the operations and interface available at the KDC, as well as general message formats for the interactions between Clients and KDC. At the same time, communications in the group can rely on different approaches, e.g., based on multicast [I-D.ietf-core-groupcomm-bis] or on publish-subscribe messaging [I-D.ietf-core-coap-pubsub], and can be protected in different ways.

Therefore, this document delegates details on the communication and security approaches used in a group to separate application profiles. These are specialized instances of this document, targeting a particular group communication approach and defining how communications in the group are protected, as well as the specific keying material and configuration parameters provided to group members.

In order to ensure consistency and aid the development of such application profiles, Appendix A of this document defines a number of related compliance requirements. In particular, Appendix A.1 compiles the requirements that application profiles are REQUIRED to fulfill; these are referred to by an identifier that starts with "REQ". Instead, Appendix A.2 compiles the requirements that application profiles MAY fulfill; these are referred to by an identifier that starts with "OPT".

New keying material is intended to be generated and distributed to the group upon membership changes (rekeying). If the application requires backward security (i.e., new group members must be prevented from accessing communications in the group prior to their joining), then a rekeying has to occur every time new members join the group. If the application requires forward security (i.e., former group members must be prevented from accessing communications in the group after their leaving), then a rekeying has to occur every time current members leave or are evicted from the group.

A group rekeying scheme performs the actual distribution of the new keying material, by rekeying the current group members when a new Client joins the group, and the remaining group members when a Client leaves the group. This can rely on different approaches, including efficient group rekeying schemes such as [RFC2093], [RFC2094], and [RFC2627].

Consistently with what is recommended in the ACE framework, this document uses CBOR [RFC8949] for data encoding. However, using JSON [RFC8259] instead of CBOR is possible, by relying on the conversion method specified in Sections 6.1 and 6.2 of [RFC8949].

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 [RFC9200] and in the Authorization Information Format (AIF) [RFC9237] to express authorization information. 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).

* The terms and concepts described in CoAP [RFC7252]. Unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol".

* The terms and concepts described in CDDL [RFC8610], CBOR [RFC8949], and COSE [RFC9052][RFC9053][RFC9338].

A node interested to participate in group communication as well as already participating as a group member is interchangeably denoted as "Client".

This document also uses the following terms.

* Group: a set of nodes that share common keying material and security parameters used to protect their communications with one another. That is, the term refers to a "security group".

  This term is not to be confused with an "application group", which has relevance at the application level and whose members share a common pool of resources or content. Examples of application groups are the set of all nodes deployed in a same physical room, or the set of nodes registered to a pub-sub topic.

  This term is also not to be confused with a "multicast group", which has relevance at the network level and whose members all listen to a group network address for receiving messages sent to that group. An example of multicast group is the set of nodes that are configured to receive messages that are sent to the group's associated IP multicast address.

  The same security group might be associated with multiple application groups. Also, the same application group can be associated with multiple security groups. Further details and considerations on the mapping between the three types of group are out of the scope of this document.

* Key Distribution Center (KDC): the entity responsible for managing one or multiple groups, with particular reference to the group membership and the keying material to use for protecting group communications.

Furthermore, this document uses "names" or "identifiers" for groups and nodes. Their different meanings are summarized below.

* Group name: The identifier of a group, as a text string encoded as UTF-8 [RFC3629]. Once established, it is invariant. It is used in the interactions between Client, AS, and RS to identify a group. A group name is always unique among the group names of the existing groups under the same KDC.

* GROUPNAME: The text string used in URIs to identify a group. Once established, it is invariant. GROUPNAME uniquely maps to the group name of a group, although they do not necessarily coincide.

* Group identifier: the identifier of the group keying material used in a group. Unlike group name and GROUPNAME, this identifier changes over time, when the group keying material is updated.

* Node name: The identifier of a node, as a text string encoded as UTF-8 [RFC3629] and consistent with the semantics of URI path segments (see Section 3.3 of [RFC3986]). Once established, it is invariant. It is used in the interactions between Client and RS, as well as to identify a member of a group. A node name is always unique among the node names of the current nodes within a group.

* NODENAME: The text string used in URIs to identify a member of a group. Once established, it is invariant. Its value coincides with the node name of the associated group member.

This document additionally uses the following terminology:

* Transport profile: a profile of ACE as per Section 5.8.4.3 of [RFC9200]. A transport profile specifies the communication protocol and communication security protocol between an ACE Client and Resource Server, as well as proof-of-possession methods, if it supports proof-of-possession access tokens, etc. Transport profiles of ACE include, for instance, [RFC9203], [RFC9202], and [RFC9431].

* Application profile: a profile that defines how applications enforce and use supporting security services they require. These services may include, for instance, provisioning, revocation, and distribution of keying material. An application profile may define specific procedures and message formats.

* Authentication credential: the set of information associated with an entity, including that entity's public key and parameters associated with the public key. Examples of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC5280], and C509 certificates [I-D.ietf-cose-cbor-encoded-cert].

* Individual keying material: information pertaining exclusively to a group member, as associated with its group membership and related to other keying material and parameters used in the group. For example, this can be an identifier that the secure communication protocol employs to uniquely identify a node as a group member (e.g., a cryptographic key identifier uniquely associated with the group member in question). The specific nature and format of individual keying material used in a group is defined in application profiles of this specification. The individual keying material of a group member is not related to the secure association between that group member and the KDC.

Examples throughout this document are expressed in CBOR diagnostic notation without the tag and value abbreviations.

2. Overview

At a high level, the key provisioning process is separated in two phases: the first one follows the ACE Framework between Client, AS, and KDC; the second one is the actual key distribution between Client and KDC. After the two phases are completed, the Client is able to participate in the group communication, via a Dispatcher entity.

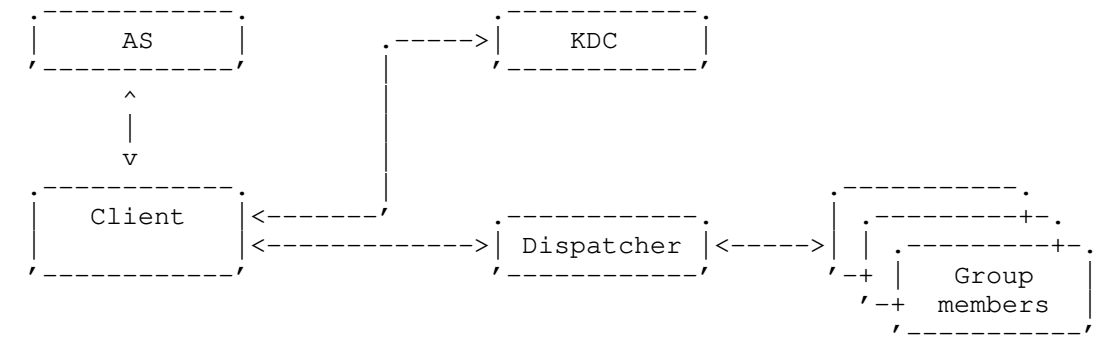


Figure 1: Key Distribution Participants

The following participants (see Figure 1) take part in the authorization and key distribution.

* Client (C): node that wants to join a group and take part in group communication with other group members. Within the group, the Client can have different roles.

* Authorization Server (AS): as per the AS defined in the ACE Framework [RFC9200], it enforces access policies that prescribe whether a node is allowed to join a given group and with what roles and rights (e.g., write and/or read).

* Key Distribution Center (KDC): entity that maintains the keying material to protect group communications, and provides it to Clients authorized to join a given group. During the first part of the exchange (Section 3), the KDC takes the role of the RS in the ACE Framework. During the second part (Section 4), which is not based on the ACE Framework, the KDC distributes the keying material. In addition, the KDC provides the latest keying material to group members when requested or, if required by the application, when group membership changes.

* Dispatcher: entity through which the Clients communicate with the group when sending a message intended for multiple group members. That is, the Dispatcher distributes such a one-to-many message to the group members as intended recipients. The Dispatcher does not have access to the group keying material. A single-recipient message intended for only one group member may be delivered by alternative means, with no assistance from the Dispatcher.

  Examples of a Dispatcher are: the Broker in a pub-sub setting; a relayer for group communication that delivers group messages as multiple unicast messages to all group members; an implicit entity as in a multicast communication setting, where messages are transmitted to a multicast IP address and delivered on the transport channel.

  If it consists of an explicit entity such as a pub-sub Broker or a message relayer, the Dispatcher is comparable to an untrusted on-path intermediary, and as such it is able to see the messages sent by Clients in the group, but not to decrypt them and read their plain content.

This document specifies a mechanism for:

* Authorizing a Client to join the group (Section 3), and providing it with the group keying material to communicate with the other group members (Section 4).

* Allowing a group member to retrieve group keying material (Section 4.3.2.1 and Section 4.8.1.1).

* Allowing a group member to retrieve authentication credentials of other group members (Section 4.4.1.1) and to provide an updated authentication credential (Section 4.9.1.1).

* Allowing a group member to leave the group (Section 4.8.3.1).

* Evicting a group member from the group (Section 5).

* Renewing and re-distributing the group keying material (rekeying), e.g., upon a membership change in the group (Section 6).

  Rekeying the group may result in a temporary misalignment of the keying material stored by the different group members. Different situations where this can happen and how they can be handled are discussed in Section 6.3.

Figure 2 provides a high level overview of the message flow for a node joining a group. The message flow can be expanded as follows.

1. The joining node requests an access token from the AS, in order to access one or more group-membership resources at the KDC and hence join the associated groups.

   This exchange between Client and AS MUST be secured, as specified by the transport profile of ACE used between Client and KDC. Based on the response from the AS, the joining node will establish or continue using a secure communication association with the KDC.

2. The joining node transfers authentication and authorization information to the KDC, by transferring the obtained access token. This is typically achieved by including the access token in a request sent to the /authz-info endpoint at the KDC.

   Once this exchange is completed, the joining node MUST have a secure communication association established with the KDC, before joining a group under that KDC.

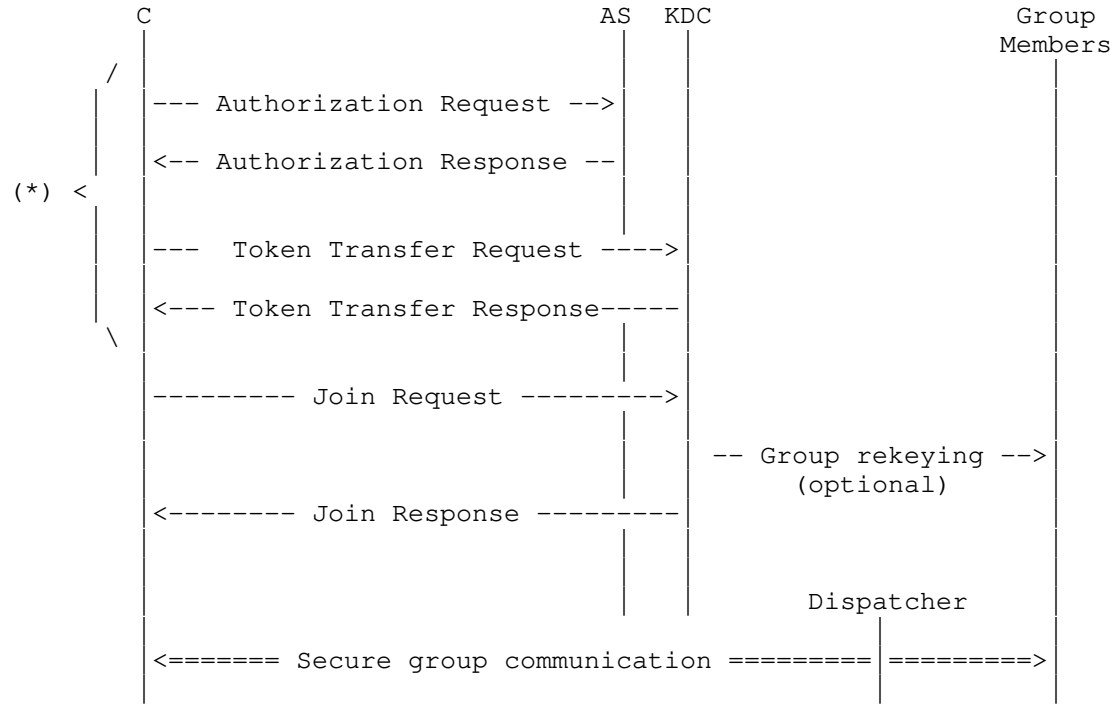
   This exchange and the following secure communications between the Client and the KDC MUST occur in accordance with the transport profile of ACE used between Client and KDC, such as the DTLS transport profile [RFC9202] and OSCORE transport profile [RFC9203] of ACE.

3. The joining node starts the joining process to become a member of the group, by sending a request to the related group-membership resource at the KDC. Based on the application requirements and policies, the KDC may perform a group rekeying, by generating new group keying material and distributing it to the current group members through the rekeying scheme used in the group.

   At the end of the joining process, the joining node has received from the KDC the parameters and group keying material to securely communicate with the other group members. Also, the KDC has stored the association between the authorization information from the access token and the secure session with the joining node.

4. The joining node and the KDC maintain the secure association, to support possible future communications. These especially include key management operations, such as retrieval of updated keying material or participation to a group rekeying process.

5. The joining node can communicate securely with the other group members, using the keying material provided in step 3.



(*) Defined in the ACE framework

Figure 2: Message Flow Upon New Node's Joining

3. Authorization to Join a Group

This section describes in detail the format of messages exchanged by the participants when a node requests access to a given group. This exchange is based on ACE [RFC9200].

As defined in [RFC9200], the Client asks the AS for the authorization to join the group through the KDC (see Section 3.1). If the request is approved and authorization is granted, the AS provides the Client with a proof-of-possession access token and parameters to securely communicate with the KDC (see Section 3.2).

Communications between the Client and the AS MUST be secured, according to what is defined by the used transport profile of ACE. The Content-Format used in the message also depends on the used transport profile of ACE. For example, it can be application/ace+cbor for the first two messages and application/cwt for the third message, which are defined in the ACE framework.

The transport profile of ACE also defines a number of details such as the communication and security protocols used with the KDC (see Appendix C of [RFC9200]).

Figure 3 gives an overview of the exchange described above.

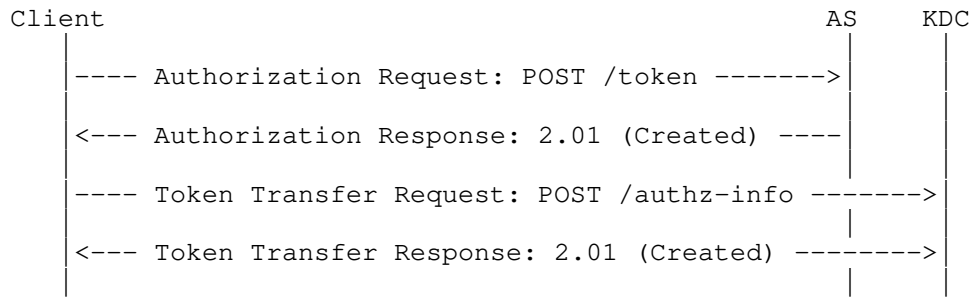


Figure 3: Message Flow of Join Authorization

3.1. Authorization Request

The Authorization Request sent from the Client to the AS is defined in Section 5.8.1 of [RFC9200] and MAY contain the following parameters, which, if included, MUST have the format and value as specified below.

* 'scope', specifying the names of the groups that the Client requests to access, and optionally the roles that the Client requests to have in those groups.

  This parameter is encoded as a CBOR byte string, which wraps a CBOR array of scope entries. All the scope entries are specified according to a same format, i.e., either the AIF format or the textual format defined below.

  - If the AIF format is used, each scope entry is encoded as per [RFC9237], i.e., as a CBOR array [Toid, Tperm]. If a scope entry expresses a set of roles to take in a group as per this document, the object identifier "Toid" specifies the group name and MUST be encoded as a CBOR text string, while the permission set "Tperm" specifies the roles that the Client wishes to take in the group.

    The AIF format is the default format for application profiles of this specification, and is preferable for those that aim for a compact encoding of scope. This is desirable especially for application profiles defining several roles, with the Client possibly asking for multiple roles combined.

    Figure 4 shows an example in CDDL notation [RFC8610] where scope uses the AIF format.

  - If the textual format is used, each scope entry is a CBOR array formatted as follows.

    o As first element, the group name, encoded as a CBOR text string.

    o Optionally, as second element, the role or CBOR array of roles that the Client wishes to take in the group. This element is optional since roles may have been pre-assigned to the Client, as associated with its verifiable identity credentials. Alternatively, the application may have defined a single, well-known role for the target resource(s) and audience(s).

    Figure 5 shows an example in CDDL notation where scope uses the textual format, with group name and role identifiers encoded as CBOR text strings.

It is REQUIRED of application profiles of this specification to specify the exact format and encoding of scope (REQ1). This includes defining the set of possible roles and their identifiers, as well as the corresponding encoding to use in the scope entries according to the used scope format.

If the application profile uses the AIF format, it is also REQUIRED to register its specific instance of "Toid" and "Tperm", as well as the corresponding Media Type and Content-Format, as per the guidelines in [RFC9237] (REQ2).

If the application profile uses the textual format, it MAY additionally specify CBOR values to use for abbreviating the role identifiers (OPT1).

* 'audience', with an identifier of the KDC.

As defined in [RFC9200], other additional parameters can be included if necessary.

```
;# include rfc9237

gname = tstr

permissions = uint .bits roles

roles = &(
   Requester: 1,
   Responder: 2,
   Monitor: 3,
   Verifier: 4
)

scope_entries = AIF-Generic<gname, permissions>

scope = bstr .cbor scope_entries
```

Figure 4: Example of scope using the AIF format

```
gname = tstr

role = tstr

scope_entry = [ gname , ? ( role / [ 2*role ] ) ]

scope_entries = [ * scope_entry ]

scope = bstr .cbor scope_entries
```

Figure 5: Example of scope using the textual format, with the role identifiers encoded as text strings

3.2. Authorization Response

The AS processes the Authorization Request as defined in Section 5.8.2 of [RFC9200], especially verifying that the Client is authorized to access the specified groups with the requested roles, or possibly a subset of those.

In case of successful verification, the Authorization Response sent from the AS to the Client is also defined in Section 5.8.2 of [RFC9200]. Note that the parameter 'expires_in' MAY be omitted if the application defines how the expiration time is communicated to the Client via other means, or if it establishes a default value.

Additionally, when included, the following parameter MUST have the corresponding values:

* 'scope' has the same format and encoding of 'scope' in the Authorization Request, defined in Section 3.1. If this parameter is not present, the granted scope is equal to the one requested in Section 3.1.

The proof-of-possession access token (in 'access_token' above) MUST contain the following parameters:

* a confirmation claim (see, for example 'cnf' defined in Section 3.1 of [RFC8747] for CWT);

* an expiration time claim (see, for example 'exp' defined in Section 3.1.4 of [RFC8392] for CWT);

* a scope claim (see, for example 'scope' registered in Section 8.14 of [RFC9200] for CWT).

  This claim specifies the same access control information as in the 'scope' parameter of the Authorization Response, if the parameter is present in the message. If the parameter is not present, the claim specifies the access control information as in the 'scope' parameter of the Authorization Request, if present, or the default scope that the AS is granting the Client, if not present.

  By default, this claim has the same encoding as the 'scope' parameter in the Authorization Request, defined in Section 3.1.

Optionally, an alternative extended format of scope defined in Section 7 can be used. This format explicitly signals the semantics used to express the actual access control information, and according to which this has to be parsed. This enables a Resource Server to correctly process a received access token, also in case:

- The Resource Server implements a KDC that supports multiple application profiles of this specification, using different scope semantics; and/or

- The Resource Server implements further services beyond a KDC for group communication, using different scope semantics.

If the Authorization Server is aware that this applies to the Resource Server for which the access token is issued, the Authorization Server SHOULD use the extended format of scope defined in Section 7.

The access token MAY additionally contain other claims that the transport profile of ACE requires, or other optional parameters.

When receiving an Authorization Request from a Client that was previously authorized, and for which the AS still stores a valid non-expired access token, the AS MAY reply with that token. Note that it is up to application profiles of ACE to make sure that re-posting the same access token does not cause re-use of keying material between nodes (for example, that is accomplished with the use of random nonces in [RFC9203]).

3.3. Token Transferring

The Client sends a Token Transfer Request to the KDC, i.e., a CoAP POST request including the access token and targeting the authz-info endpoint (see Section 5.10.1 of [RFC9200]).

Note that this request deviates from the one defined in [RFC9200], since it allows asking the KDC for additional information concerning the authentication credentials used in the group to ensure source authentication, as well as for possible additional group parameters.

The joining node MAY ask for this information from the KDC through the same Token Transfer Request. In this case, the message MUST have Content-Format set to application/ace+cbor defined in Section 8.16 of [RFC9200], and the message payload MUST be formatted as a CBOR map, which MUST include the access token. The CBOR map MAY additionally include the following parameter, which, if included, MUST have the format and value as specified below.

* 'sign_info' defined in Section 3.3.1, specifying the CBOR simple value "null" (0xf6) to request information about the signature algorithm, the signature algorithm parameters, the signature key parameters, and the exact format of authentication credentials used in the groups that the Client has been authorized to join.

Alternatively, such information may be pre-configured on the joining node, or may be retrieved by alternative means. For example, the joining node may have performed an early group discovery process and obtained the link to the associated group-membership resource at the KDC, together with attributes descriptive of the group configuration (see, e.g., [I-D.tiloca-core-oscore-discovery]).

After successful verification, the Client is authorized to receive the group keying material from the KDC and join the group. Hence, the KDC replies to the Client with a Token Transfer Response, i.e., a CoAP 2.01 (Created) response.

The Token Transfer Response MUST have Content-Format "application/ace+cbor", and its payload is a CBOR map. Note that this deviates from what is defined in the ACE framework, where the response from the authz-info endpoint is defined as conveying no payload (see Section 5.10.1 of [RFC9200]).

If a scope entry in the access token specifies a role that requires the Client to send its own authentication credential to the KDC when joining the related group, then the CBOR map MUST include the parameter 'kdcchallenge' defined in Section 3.3.2, specifying a dedicated challenge N_S generated by the KDC.

Later, when joining the group (see Section 4.3.1.1), the Client uses the 'kdcchallenge' value and additional information to build a proof-of-possession (PoP) input. This is in turn used to compute a PoP evidence, which the Client also provides to the KDC in order to prove possession of its own private key (see the 'client_cred_verify' parameter in Section 4.3.1).

While storing the access token, the KDC MUST store the 'kdcchallenge' value associated with the Client at least until it receives a Join Request from the Client (see Section 4.3.1.1), to be able to verify the PoP evidence provided during the join process, and thus that the Client possesses its own private key. The KDC deletes the 'kdcchallenge' value associated with the Client upon deleting the access token (e.g., upon its expiration, see Section 5.10.3 of [RFC9200]).

The same 'kdcchallenge' value MAY be reused several times by the Client, to generate a new PoP evidence, e.g., in case the Client provides the KDC with a new authentication credential while being a group member (see Section 4.9.1.1), or joins a different group where it intends to use a different authentication credential. Therefore, it is RECOMMENDED that the KDC keeps storing the 'kdcchallenge' value after the first join is processed as well. If, upon receiving a Join Request from a Client, the KDC has already discarded the 'kdcchallenge' value, that will trigger an error response with a newly generated 'kdcchallenge' value that the Client can use to restart the join process, as specified in Section 4.3.1.1.

If 'sign_info' is included in the Token Transfer Request, the KDC SHOULD include the 'sign_info' parameter in the Token Transfer Response, as per the format defined in Section 3.3.1. Note that the field 'id' of each 'sign_info_entry' specifies the name, or array of group names, to which that 'sign_info_entry' applies. As an exception, the KDC MAY omit the 'sign_info' parameter in the Token Transfer Response even if 'sign_info' is included in the Token Transfer Request, in case none of the groups that the Client is authorized to join uses signatures to achieve source authentication.

Note that the CBOR map specified as payload of the 2.01 (Created) response may include further parameters, e.g., according to the used transport profile of ACE. Application profiles of this specification MAY define additional parameters to use within this exchange (OPT2).

Application profiles of this specification MAY define alternative specific negotiations of parameter values for the signature algorithm and signature keys, if 'sign_info' is not used (OPT3).

If allowed by the used transport profile of ACE, the Client may provide the Access Token to the KDC by other means than the Token Transfer Request. An example is the DTLS transport profile of ACE, where the Client can provide the access token to the KDC during the secure session establishment (see Section 3.3.2 of [RFC9202]).

3.3.1. 'sign_info' Parameter

The 'sign_info' parameter is an OPTIONAL parameter of the request and response messages exchanged between the Client and the authz-info endpoint at the RS (see Section 5.10.1. of [RFC9200]).

This parameter allows the Client and the RS to exchange information about a signature algorithm and about authentication credentials to accordingly use for signature verification. Its exact semantics and content are application specific.

In this specification and in application profiles building on it, this parameter is used to exchange information about the signature algorithm and about authentication credentials to be used with it, in the groups indicated by the transferred access token as per its 'scope' claim (see Section 3.2).

When used in the Token Transfer Request sent to the KDC (see Section 3.3), the 'sign_info' parameter specifies the CBOR simple value "null" (0xf6). This is done to ask for information about the signature algorithm and about the authentication credentials used in the groups that, as per the granted roles, the Client has been authorized to join or interact with (e.g., as an external signature verifier).

When used in the following Token Transfer Response from the KDC (see Section 3.3), the 'sign_info' parameter is a CBOR array of one or more elements. The number of elements is at most the number of groups that the Client has been authorized to join or interact with. Each element contains information about signing parameters and about authentication credentials for one or more groups, and is formatted as follows.

* The first element 'id' is a group name or a CBOR array of group names, associated with groups for which the next four elements apply. Each specified group name is a CBOR text string and is hereafter referred to as 'gname'.

* The second element 'sign_alg' is a CBOR integer or a text string, indicating the signature algorithm used in the groups identified by the 'gname' values. It is REQUIRED of application profiles to define specific values that this parameter can take (REQ3), selected from the set of signing algorithms of the COSE Algorithms registry [COSE.Algorithms].

* The third element 'sign_parameters' is a CBOR array indicating the parameters of the signature algorithm used in the groups identified by the 'gname' values. Its content depends on the value of 'sign_alg'. It is REQUIRED of application profiles to define the possible values and structure for the elements of this parameter (REQ4).

* The fourth element 'sign_key_parameters' is a CBOR array indicating the parameters of the key used with the signature algorithm, in the groups identified by the 'gname' values. Its content depends on the value of 'sign_alg'. It is REQUIRED of application profiles to define the possible values and structure for the elements of this parameter (REQ5).

* The fifth element 'cred_fmt' is either a CBOR integer indicating the format of authentication credentials used in the groups identified by the 'gname' values, or has value the CBOR simple value "null" (0xf6) indicating that the KDC does not act as repository of authentication credentials for group members. Its acceptable integer values are taken from the 'Label' column of the "COSE Header Parameters" registry [COSE.Header.Parameters], with some of those values also indicating the type of container to use for exchanging the authentication credentials with the KDC (e.g., a chain or bag of certificates). It is REQUIRED of application profiles to define specific values to use for this parameter, consistently with the acceptable formats of authentication credentials (REQ6).

The CDDL notation [RFC8610] of the 'sign_info' parameter is given below.

```
sign_info = sign_info_req / sign_info_resp

sign_info_req  = null                   ; in the Token Transfer
                                        ; Request to the KDC

sign_info_resp = [ + sign_info_entry ] ; in the Token Transfer
                                        ; Response from the KDC

sign_info_entry =
[
  id : gname / [ + gname ],
  sign_alg : int / tstr,
  sign_parameters : [ any ],
  sign_key_parameters : [ + parameter : any ],
  cred_fmt : int / null
]

gname = tstr
```

This format is consistent with every signature algorithm currently defined in [RFC9053], i.e., with algorithms that have only the COSE key type as their COSE capability. Appendix B describes how the format of each 'sign_info_entry' can be generalized for possible future registered algorithms having a different set of COSE capabilities.

3.3.2. 'kdcchallenge' Parameter

The 'kdcchallenge' parameter is an OPTIONAL parameter of the response message returned from the authz-info endpoint at the RS, as defined in Section 5.10.1 of [RFC9200]. This parameter contains a challenge generated by the RS and provided to the Client.

In this specification and in application profiles building on it, the Client can use this challenge to prove possession of its own private key in the Join Request (see the 'client_cred_verify' parameter in Section 4.3.1).

4. KDC Functionalities

This section describes the functionalities provided by the KDC, as related to the provisioning of the keying material as well as to the group membership management.

In particular, this section defines the interface available at the KDC; specifies the handlers of each resource provided by the KDC interface; and describes how Clients interact with those resources to join a group and to perform additional operations as group members.

A key operation that the Client can perform after transferring the access token to the KDC is a Join Request-Response exchange with the KDC. In the Join Request, the Client specifies the group it requests to join (see Section 4.3.1.1). The KDC will then verify the access token and that the Client is authorized to join the specified group. If so, the KDC provides the Client with the keying material to securely communicate with the other members of the group.

Later on as a group member, the Client can also rely on the interface at the KDC to perform additional operations, consistent with the roles it has in the group.

4.1. Interface at the KDC

The KDC provides its interface by hosting the following resources. Note that the root url-path "ace-group" used hereafter is a default name; implementations are not required to use this name, and can define their own instead.

If request messages sent to the KDC as well as success response messages from the KDC include a payload and specify a Content-Format, those messages MUST have Content-Format set to application/ace-groupcomm+cbor, defined in Section 11.2. CBOR labels for the message parameters are defined in Section 8.

* /ace-group : the path of this root resource is invariant once the resource is established, and indicates that this specification is used. If other applications run on a KDC implementing this specification and use this same path, those applications will collide, and a mechanism will be needed to differentiate the endpoints.

  A Client can access this resource in order to retrieve a set of group names, each corresponding to one of the specified group identifiers. This operation is described in Section 4.2.1.1.

  Clients may be authorized to access this resource even without being members of any group at the KDC, and even if they are not authorized to become group members (e.g., when authorized to be external signature verifiers).

  The Interface Description (if=) Link Target Attribute value "ace.groups" is registered in Section 11.5 and can be used to describe the interface provided by this root resource.

  The example below shows an exchange with a KDC with address 2001:db8::ab that hosts the resource /ace-group and returns a link to such a resource in link-format [RFC6690].

  Request:

  ```
  Header: GET (Code=0.01)
  Uri-Host: "kdc.example.com"
  Uri-Path: ".well-known"
  Uri-Path: "core"
  Uri-Query: "if=ace.groups"
  ```

  Response:

  ```
  Header: Content (Code=2.05)
  Content-Format: 40 (application/link-format)
  Payload:
    <coap://[2001:db8::ab]/ace-group>;if="ace.groups"
  ```

* /ace-group/GROUPNAME : one such sub-resource to /ace-group is hosted for each group with name GROUPNAME that the KDC manages. In particular, it is the group-membership resource associated with that group, of which it contains the symmetric group keying material.

A Client can access this resource in order to join the group with name GROUPNAME, or later as a group member to retrieve the current group keying material. These operations are described in Section 4.3.1.1 and Section 4.3.2.1, respectively.

The Interface Description (if=) Link Target Attribute value "ace.group" is registered in Section 11.5 and can be used to describe the interface provided by a group-membership resource.

The example below shows an exchange with a KDC with address 2001:db8::ab that hosts the group-membership resource /ace-group/gp1 and returns a link to such a resource in link-format [RFC6690].

Request:

```
Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: ".well-known"
Uri-Path: "core"
Uri-Query: "if=ace.group"
```

Response:

```
Header: Content (Code=2.05)
Content-Format: 40 (application/link-format)
Payload:
  <coap://[2001:db8::ab]/ace-group/gp1>;if="ace.group"
```

If it is not required that the value of the GROUPNAME URI path and the group name in the access token scope ('gname' in Section 3.2) coincide, the KDC MUST implement a mechanism to map the GROUPNAME value in the URI to the group name, in order to refer to the correct group (REQ7).

* /ace-group/GROUPNAME/creds : the path of this resource is invariant once the resource is established. This resource contains the authentication credentials of all the members of the group with name GROUPNAME.

  This resource is created only in case the KDC acts as a repository of authentication credentials for group members.

As a group member, a Client can access this resource in order to retrieve the authentication credentials of other group members. That is, the Client can retrieve the authentication credentials of all the current group members, or a subset of them by specifying filter criteria. These operations are described in Section 4.4.2.1 and Section 4.4.1.1, respectively.

Clients may be authorized to access this resource even without being group members, e.g., if authorized to be external signature verifiers for the group.

* /ace-group/GROUPNAME/kdc-cred : the path of this resource is invariant once the resource is established. This resource contains the authentication credential of the KDC for the group with name GROUPNAME.

  This resource is created only in case the KDC has an associated authentication credential and this is required for the correct group operation. It is REQUIRED of application profiles to define whether the KDC has such an associated authentication credential (REQ8).

  As a group member, a Client can access this resource in order to retrieve the current authentication credential of the KDC.

  Clients may be authorized to access this resource even without being group members, e.g., if authorized to be external signature verifiers for the group.

* /ace-group/GROUPNAME/policies : the path of this resource is invariant once the resource is established. This resource contains the group policies of the group with name GROUPNAME.

  A Client can access this resource as a group member in order to retrieve the group policies. This operation is described in Section 4.6.1.1.

* /ace-group/GROUPNAME/num : the path of this resource is invariant once the resource is established. This resource contains the current version number for the symmetric group keying material of the group with name GROUPNAME.

  A Client can access this resource as a group member in order to retrieve the version number of the keying material currently used in the group. This operation is described in Section 4.7.1.1.

* /ace-group/GROUPNAME/nodes/NODENAME : one such sub-resource of /ace-group/GROUPNAME is hosted for each group member of the group with name GROUPNAME. Each such resource is identified by the node name NODENAME of the associated group member, and contains the group keying material and the individual keying material for that group member.

  A Client as a group member can access this resource in order to retrieve the current group keying material together with its individual keying material; request new individual keying material to use in the group; and leave the group. These operations are described in Section 4.8.1.1, Section 4.8.2.1, and Section 4.8.3.1, respectively.

* /ace-group/GROUPNAME/nodes/NODENAME/cred : the path of this resource is invariant once the resource is established. This resource contains the individual authentication credential for the node with name NODENAME, as group member of the group with name GROUPNAME.

  A Client can access this resource in order to upload at the KDC a new authentication credential to use in the group. This operation is described in Section 4.9.1.1.

  This resource is not created if the group member does not have an authentication credential to use in the group, or if the KDC does not store the authentication credentials of group members.

The KDC is expected to fully provide the interface defined above. It is otherwise REQUIRED of the application profiles of this specification to indicate which resources are not hosted, i.e., which parts of the interface defined in this section are not supported by the KDC (REQ9). Application profiles of this specification MAY extend the KDC interface, by defining additional handlers, as well as defining additional resources and their handlers.

It is REQUIRED of application profiles of this specification to register a Resource Type for the group-membership resource (REQ10), i.e., the group-membership resource at /ace-group/GROUPNAME. This Resource Type can be used to discover the correct URL for sending a Join Request to the KDC. This Resource Type can also be used to indicate which specific application profile of this specification is used by a specific group-membership resource at the KDC.

It is REQUIRED of application profiles of this specification to define what specific actions (e.g., CoAP methods) are allowed on each resource provided by the KDC interface, depending on whether the Client is a current group member; the roles that a Client is

authorized to take as per the obtained access token (see Section 3.1); and the roles that the Client has as current group member (REQ11).

4.1.1. Operations Supported by Clients

It is expected that a Client minimally supports the following set of primary operations and corresponding interactions with the KDC.

* FETCH request to /ace-group/ , in order to retrieve group names associated with group identifiers.

* POST and GET requests to /ace-group/GROUPNAME/ , in order to join a group (POST) and later retrieve the current group key material as a group member (GET).

* GET and FETCH requests to /ace-group/GROUPNAME/creds , in order to retrieve the authentication credentials of all the other group members (GET) or only some of them by filtering (FETCH). While retrieving authentication credentials remains possible by using GET requests, retrieval by filtering allows Clients to greatly limit the size of exchanged messages.

* GET request to /ace-group/GROUPNAME/num , in order to retrieve the current version of the group key material as a group member.

* DELETE request to /ace-group/GROUPNAME/nodes/NODENAME , in order to leave the group.

In addition, some Clients may rather not support the following set of secondary operations and corresponding interactions with the KDC. This can be specified, for instance, in compliance documents defining minimalistic Clients and their capabilities in specific deployments. In turn, these might also have to consider the used application profile of this specification.

* GET request to /ace-group/GROUPNAME/kdc-cred , in order to retrieve the current authentication credential of the KDC. This is relevant only if the KDC has an associated authentication credential and this is required for the correct group operation.

* GET request to /ace-group/GROUPNAME/policies , in order to retrieve the current group policies as a group member.

* GET request to /ace-group/GROUPNAME/nodes/NODENAME , in order to retrieve the current group keying material and individual keying material. The former can also be retrieved through a GET request to /ace-group/GROUPNAME/ (see above).

* PUT request to /ace-group/GROUPNAME/nodes/NODENAME , in order to ask for new individual keying material. Alternatively, the Client could obtain new individual keying material by re-joining the group through a POST request to /ace-group/GROUPNAME/ (see above). Furthermore, depending on its roles in the group or on the application profile of this specification, the Client might simply not be associated with any individual keying material.

* POST request to /ace-group/GROUPNAME/nodes/NODENAME/cred , in order to provide the KDC with a new authentication credential. Alternatively, the Client could provide a new authentication credential by re-joining the group through a POST request to /ace-group/GROUPNAME/ (see above). Furthermore, depending on its roles in the group, the Client might simply not have an associated authentication credential to provide.

It is REQUIRED of application profiles of this specification to categorize possible newly defined operations for Clients into primary operations and secondary operations, and to provide accompanying considerations (REQ12).

4.1.2. Error Handling

Upon receiving a request from a Client, the KDC MUST check that it is storing a valid access token from that Client. If this is not the case, the KDC MUST reply with a 4.01 (Unauthorized) error response.

Unless the request targets the /ace-group resource, the KDC MUST check that it is storing a valid access token from that Client such that:

* The scope specified in the access token includes a scope entry related to the group name GROUPNAME associated with the targeted resource; and

* The set of roles specified in that scope entry allows the Client to perform the requested operation on the targeted resource (REQ11).

In case the KDC stores a valid access token but the verifications above fail, the KDC MUST reply with a 4.03 (Forbidden) error response. This response MAY be an AS Request Creation Hints, as defined in Section 5.3 of [RFC9200], in which case the Content-Format MUST be set to application/ace+cbor.

If the request is not formatted correctly (e.g., required fields are not present or are not encoded as expected), the handler MUST reply with a 4.00 (Bad Request) error response.

If the request includes unknown or non-expected fields, the handler MUST silently ignore them and continue processing the request. Application profiles of this specification MAY define optional or mandatory payload formats for specific error cases (OPT4).

Some error responses from the KDC can convey error-specific information according to the problem-details format defined 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-groupcomm-error' registered in Section 11.6 of this document.

  This entry is formatted as a CBOR map including only one field, namely 'error-id'. The map key for 'error-id' is the CBOR unsigned integer with value 0. The value of 'error-id' is a CBOR integer specifying the error occurred at the KDC. This value is taken from the 'Value' column of the "ACE Groupcomm Errors" registry defined in Section 11.12 of this document.

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

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

* It MAY include further Standard Problem Detail entries or Custom Problem Detail entries (see [RFC9290]).

  In particular, it can include the Standard Problem Detail entry 'detail' (map key -2), whose value is a CBOR text string that specifies a human-readable, diagnostic description of the error occurred at the KDC. The diagnostic text is intended for software engineers as well as for device and network operators, in order to aid debugging and provide context for possible intervention. The diagnostic message SHOULD be logged by the KDC. The 'detail' entry is unlikely relevant in an unattended setup where human intervention is not expected.

An example of error response using the problem-details format is shown in Figure 6.

Response:

```
Header: Service Unavailable (Code=5.03)
Content-Format: application/concise-problem-details+cbor
Payload:
{
  / title /                -1: "No available node identifiers",
  / detail /               -2: "Things will change after a
                                group rekeying; try later",
  / ace-groupcomm-error /  TBD: {
    / error-id /  0: 4 / "No available node identifiers" /,
  }
}
```

Figure 6: Example of Error Response with Problem Details

Note to RFC Editor: In the figure above, please replace "TBD" with the unsigned integer assigned as key value to the Custom Problem Detail entry 'ace-groupcomm-error' (see Section 11.6). Then, please delete this paragraph.

The problem-details format in general and the Custom Problem Detail entry 'ace-groupcomm-error' in particular are OPTIONAL to support for Clients. A Client supporting the entry 'ace-groupcomm-error' and able to understand the specified error may use that information to determine what actions to take next.

Section 9 of this specification defines an initial set of error identifiers, as possible values for the 'error-id' field. Application profiles of this specification inherit this initial set of error identifiers and MAY define additional values (OPT5).

4.2. /ace-group

This resource implements the FETCH handler.

4.2.1. FETCH Handler

The FETCH handler receives group identifiers and returns the corresponding group names and GROUPNAME URIs.

The handler expects a request with payload formatted as a CBOR map, which MUST contain the following fields:

* 'gid', whose value is encoded as a CBOR array, containing one or more group identifiers. The exact encoding of the group identifier MUST be specified by the application profile (REQ13). The Client indicates that it wishes to receive the group names of all the groups having these identifiers.

The handler identifies the groups where communications are secured by using the keying material identified by those group identifiers.

If all verifications succeed, the handler replies with a 2.05 (Content) response, whose payload is formatted as a CBOR map that MUST contain the following fields:

* 'gid', whose value is encoded as a CBOR array, containing zero or more group identifiers. The handler indicates that those are the identifiers it is sending group names for. This CBOR array is a subset of the 'gid' array in the FETCH request.

* 'gname', whose value is encoded as a CBOR array, containing zero or more group names. The elements of this array are encoded as text strings. Each element of index i in this CBOR array is associated with the element of index i in the 'gid' array.

* 'guri', whose value is encoded as a CBOR array, containing zero or more URIs, each indicating a group-membership resource. The elements of this array are encoded as text strings. Each element of index i in this CBOR array is associated with the element of index i in the 'gid' array.

If the KDC does not find any group associated with the specified group identifiers, the handler returns a response with payload formatted as a CBOR byte string of zero length.

Note that the KDC only verifies that the node is authorized by the AS to access this resource. Nodes that are not members of the group but are authorized to do signature verification on the group messages may be allowed to access this resource, if the application needs it.

4.2.1.1. Retrieve Group Names

In case the joining node only knows the group identifier of the group it wishes to join or about which it wishes to get updated information from the KDC, the node can contact the KDC to request the corresponding group name and group-membership resource URI. The node can request several group identifiers at once. It does so by sending a CoAP FETCH request to the /ace-group endpoint at the KDC formatted as defined in Section 4.2.1.

Figure 7 gives an overview of the exchanges described above, and Figure 8 shows an example.

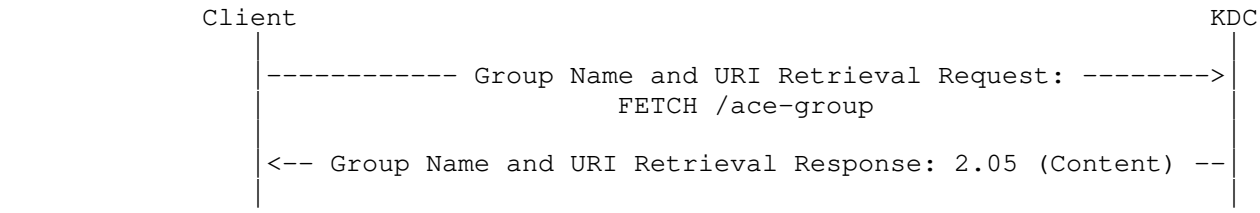


Figure 7: Message Flow of Group Name and URI Retrieval Request-Response

Request:

```
Header: FETCH (Code=0.05)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
  { "gid": [1, 2] }
```

Response:

```
Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
  { "gid": [1, 2], "gname": ["group1", "group2"],
    "guri": ["/ace-group/g1", "/ace-group/g2"] }
```

Figure 8: Example of Group Name and URI Retrieval Request-Response

4.3. /ace-group/GROUPNAME

This resource implements the POST and GET handlers.

4.3.1. POST Handler

The POST handler processes the Join Request sent by a Client to join a group, and returns a Join Response as successful result of the joining process (see Section 4.3.1.1). At a high level, the POST handler adds the Client to the list of current group members, adds the authentication credential of the Client to the list of the group members' authentication credentials, and returns the symmetric group keying material for the group identified by GROUPNAME.

The handler expects a request with payload formatted as a CBOR map, which MAY contain the following fields, which, if included, MUST have the format and value as specified below.

* 'scope', with value the specific group that the Client is attempting to join, i.e., the group name, and the roles it wishes to have in the group. This value is a CBOR byte string wrapping one scope entry, as defined in Section 3.1.

* 'get_creds', if the Client wishes to receive the authentication credentials of the current group members from the KDC. This parameter may be included in the Join Request if the KDC stores the authentication credentials of the group members, while it is not useful to include it if the Client obtains those authentication credentials through alternative means, e.g., from the AS. Note that including this parameter might result in a following Join Response of large size, which can be inconvenient for resource-constrained devices.

  If the Client wishes to retrieve the authentication credentials of all the current group members, the 'get_creds' parameter MUST encode the CBOR simple value "null" (0xf6). Otherwise, if the Client wishes to retrieve the authentication credentials of nodes with specific roles, the 'get_creds' parameter MUST encode a non-empty CBOR array, containing the three elements 'inclusion_flag', 'role_filter', and 'id_filter' as defined below.

  - The first element, namely 'inclusion_flag', encodes the CBOR simple value "true" (0xf5) if the Client wishes to receive the authentication credentials of the nodes having their node identifier specified in 'id_filter' (i.e., selection by inclusive filtering). Instead, this element encodes the CBOR simple value "false" (0xf4) if the Client wishes to receive the authentication credentials of the nodes not having the node identifiers specified in the third element 'id_filter' (i.e., selection by exclusive filtering). In the Join Request, this parameter encodes the CBOR simple value "true" (0xf5).

  - The second element, namely 'role_filter', is a CBOR array. Each element of the array contains one role or a combination of roles for the group identified by GROUPNAME. This parameter indicates that the Client wishes to receive the authentication credentials of all the group members having any of the specified roles or combination of roles (i.e., having any of those single roles, or at least all the roles indicated in any of those combinations of roles).

For example, the array ["role1", "role2+role3"] indicates that the Client wishes to receive the authentication credentials of all group members that have at least "role1" or at least both "role2" and "role3". In the Join Request this parameter is a non-empty array.

- The third element, namely 'id_filter', is a CBOR array. Each element of the array contains a node identifier of a group member for the group identified by GROUPNAME. This parameter indicates that the Client wishes to receive the authentication credentials of the nodes that have or do not have the specified node identifiers, based on the value of 'inclusion_flag' (i.e., as a selection by inclusive or exclusive filtering). In the Join Request, the Client does not filter authentication credentials based on node identifiers, so this parameter is an empty array.

  In fact, when first joining the group, the Client is not expected or capable to express a filter based on node identifiers of other group members. Instead, when already a group member and sending a Join Request to re-join, the Client is not expected to include the 'get_creds' parameter in the Join Request altogether, since it can rather retrieve authentication credentials associated with specific group identifiers as defined in Section 4.4.1.1.

The CDDL definition [RFC8610] of 'get_creds' is given in Figure 9, using as example encoding: node identifier encoded as a CBOR byte string; role identifier encoded as a CBOR text string, and combination of roles encoded as a CBOR array of roles.

Note that, for this handler, 'inclusion_flag' is always set to true, the array of roles 'role_filter' is always non-empty, while the array of node identifiers 'id_filter' is always empty. However, this is not necessarily the case for other handlers using the 'get_creds' parameter.

```
inclusion_flag = bool

role = tstr
comb_role = [ 2*role ]
role_filter = [ *(role / comb_role) ]

id = bstr
id_filter = [ *id ]

get_creds = null / [ inclusion_flag, role_filter, id_filter]
```

Figure 9: CDDL definition of 'get_creds', using as example node identifier encoded as bstr and role as tstr

* 'client_cred', encoded as a CBOR byte string, whose value is the original binary representation of the Client's authentication credential. This parameter MUST be present if the KDC is managing (collecting from/distributing to the Client) the authentication credentials of the group members and the Client's role in the group will require the Client to send messages to one or more group members. It is REQUIRED of application profiles to define the specific formats that are acceptable to use for authentication credentials in the group (REQ6).

* 'cnonce', encoded as a CBOR byte string, and including a dedicated nonce N_C generated by the Client. This parameter MUST be present.

* 'client_cred_verify', encoded as a CBOR byte string. This parameter MUST be present if the 'client_cred' parameter is present and no authentication credential associated with the Client's token can be retrieved for that group.

  This parameter contains a proof-of-possession (PoP) evidence computed by the Client over the following PoP input: the scope (encoded as a CBOR byte string), concatenated with N_S (encoded as a CBOR byte string) concatenated with N_C (encoded as a CBOR byte string), where:

  - scope is the CBOR byte string either specified in the 'scope' parameter above, if present, or encoding a default scope entry that the handler is expected to know, if omitted.

  - N_S is the challenge received from the KDC in the 'kdcchallenge' parameter of the 2.01 (Created) response to the Token Transfer Request (see Section 3.3), encoded as a CBOR byte string.

  - N_C is the nonce generated by the Client and specified in the 'cnonce' parameter above, encoded as a CBOR byte string.

  An example of PoP input to compute 'client_cred_verify' using CBOR encoding is given in Figure 10.

A possible type of PoP evidence is a signature that the Client computes by using its own private key, whose corresponding public key is specified in the authentication credential carried in the 'client_cred' parameter. Application profiles of this specification MUST specify the exact approaches used to compute the PoP evidence to include in 'client_cred_verify', and MUST specify which of those approaches is used in which case (REQ14).

If the token was not provided to the KDC through a Token Transfer Request (e.g., it is used directly to validate TLS instead), it is REQUIRED of the specific application profile to define how the challenge N_S is generated (REQ15).

* 'creds_repo', which can be present if the format of the Client's authentication credential in the 'client_cred' parameter is a certificate. In such a case, this parameter has as value the URI of the certificate. This parameter is encoded as a CBOR text string. Alternative specific encodings of this parameter MAY be defined in application profiles of this specification (OPT6).

* 'control_uri', whose value is a full URI, encoded as a CBOR text string. A default url-path is /ace-group/GROUPNAME/node, although implementations can use different ones instead. The URI MUST NOT have url-path /ace-group/GROUPNAME.

  If 'control_uri' is specified in the Join Request, the Client acts as a CoAP server and hosts a resource at this specific URI. The KDC MAY use this URI to send CoAP requests to the Client (acting as CoAP server in this exchange), for example for one-to-one provisioning of new group keying material when performing a group rekeying (see Section 4.8.1.1), or to inform the Client of its removal from the group (see Section 5).

  In particular, this resource is intended for communications concerning exclusively the group identified by GROUPNAME and whose group name is specified in the 'scope' parameter, if present. If the KDC does not implement mechanisms using this resource for that group, it can ignore this parameter. Other additional functionalities of this resource MAY be defined in application profiles of this specifications (OPT7).

```
scope, N_S, and N_C expressed in CBOR diagnostic notation:
  scope = h'826667726f7570316673656e646572'
  N_S   = h'018a278f7faab55a'
  N_C   = h'25a8991cd700ac01'


scope, N_S, and N_C as CBOR encoded byte strings:
  scope = 0x4f826667726f7570316673656e646572
  N_S   = 0x48018a278f7faab55a
  N_C   = 0x4825a8991cd700ac01

PoP input:
  0x4f 826667726f7570316673656e646572
    48 018a278f7faab55a 48 25a8991cd700ac01
```

Figure 10: Example of PoP input to compute 'client_cred_verify' using CBOR encoding

If the request does not include a 'scope' field, the KDC is expected to understand what roles the Client is requesting to join the group with. For example, as per the access token, the Client might have been granted access to the group with only one role. If the KDC cannot determine which exact roles should be considered for the Client, it MUST reply with a 4.00 (Bad Request) error response.

The handler considers the scope specified in the access token associated with the Client, and checks the scope entry related to the group identified by the GROUPNAME associated with the endpoint. In particular, the handler checks whether the set of roles specified in that scope entry includes all the roles that the Client wishes to have in the group as per the Join Request. If this is not the case, the KDC MUST reply with a 4.03 (Forbidden) error response.

If the KDC manages the group members' authentication credentials, the handler checks if one is included in the 'client_cred' field. If so, the KDC retrieves the authentication credential and performs the following actions.

* If the access token was provided through a Token Transfer Request (see Section 3.3) but the KDC cannot retrieve the 'kdcchallenge' associated with this Client (see Section 3.3), the KDC MUST reply with a 4.00 (Bad Request) error response, which MUST also have Content-Format application/ace-groupcomm+cbor. The payload of the error response is a CBOR map including a newly generated 'kdcchallenge' value, which is specified in the 'kdcchallenge' parameter. The KDC MUST store the newly generated value as the 'kdcchallenge' value associated with this Client, replacing the currently stored value (if any).

* The KDC checks the authentication credential to be valid for the group identified by GROUPNAME. That is, it checks that the authentication credential has the format used in the group, is intended for the public key algorithm used in the group, and is aligned with the possible associated parameters used in the group.

  If this verification fails, the handler MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 2 ("Authentication credential incompatible with the group configuration").

* The KDC verifies the PoP evidence contained in the 'client_cred_verify' field. Application profiles of this specification MUST specify the exact approaches used to verify the PoP evidence, and MUST specify which of those approaches is used in which case (REQ14).

  If the PoP evidence does not pass verification, the handler MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 3 ("Invalid Proof-of-Possession evidence").

If no authentication credential is included in the 'client_cred' field, the handler checks if an authentication credential is already associated with the received access token and to the group identified by GROUPNAME (see also Section 4.3.1.1). Note that the same joining node may use different authentication credentials in different groups, and all those authentication credentials would be associated with the same access token.

If an eligible authentication credential for the Client is neither present in the 'client_cred' field nor retrieved from the stored ones at the KDC, it is RECOMMENDED that the handler stops the processing and replies with a 4.00 (Bad Request) error response. Application profiles MAY define alternatives (OPT8).

If, regardless of the reason, the KDC replies with a 4.00 (Bad Request) error response, the payload of the response MAY be a CBOR map. For instance, the CBOR map can include a 'sign_info' parameter formatted as 'sign_info_res' defined in Section 3.3.1, with the 'cred_fmt' element set to the CBOR simple value "null" (0xf6) if the Client sent its own authentication credential and the KDC is not set

to store authentication credentials of the group members. When the response payload is a CBOR map including such parameters, the error response has Content-Format set to application/ace-groupcomm+cbor.

If all the verifications above succeed, the KDC proceeds as follows.

First, only in case the Client is not already a group member, the handler performs the following actions:

* The handler adds the Client to the list of current members of the group.

* The handler assigns a name NODENAME to the Client, and creates a sub-resource to /ace-group/GROUPNAME at the KDC, i.e., "/ace-group/GROUPNAME/nodes/NODENAME".

* The handler associates the node identifier NODENAME with the access token and the secure communication association for the Client.

Then, the handler performs the following actions.

* If the KDC manages the group members' authentication credentials:

  - The handler associates the retrieved Client's authentication credential to the tuple composed of the node name NODENAME, the group name GROUPNAME, and the received access token.

  - The handler adds the retrieved Client's authentication credential to the stored list of authentication credentials stored for the group identified by GROUPNAME. If such list already includes an authentication credential for the Client, but a different authentication credential is specified in the 'client_cred' field, then the handler MUST replace the old authentication credential in the list with the one specified in the 'client_cred' field.

* If backward security is prescribed by application policies installed at the KDC or by the used application profile of this specification, then the KDC MUST generate new group keying material and securely distribute it to the current group members (see Section 6).

* The handler returns a successful Join Response as defined below, containing the symmetric group keying material; the group policies; and the authentication credentials of the current members of the group, if the KDC manages those and the Client requested them.

The Join Response MUST have response code 2.01 (Created) if the Client has been added to the list of group members in this join exchange (see above), or 2.04 (Changed) otherwise, i.e., if the Client is re-joining the group without having left it.

The Join Response message MUST include the Location-Path CoAP option, specifying the URI path to the sub-resource associated with the Client, i.e., "/ace-group/GROUPNAME/nodes/NODENAME".

The Join Response message MUST have Content-Format application/ace-groupcomm+cbor. The payload of the response is formatted as a CBOR map, which MUST contain the following fields and values.

* 'gkty', identifying the key type of the keying material specified in the 'key' parameter. This parameter is encoded as a CBOR integer or a CBOR text string. The set of values can be found in the "Key Type" column of the "ACE Groupcomm Key Types" registry. Implementations MUST verify that the key type matches the application profile being used, if present, as registered in the "ACE Groupcomm Key Types" registry.

* 'key', containing the keying material for the group communication, or information required to derive it.

* 'num', containing the version number of the keying material for the group communication, formatted as a CBOR integer. This is a strictly monotonic increasing field. The application profile MUST define the initial version number (REQ16).

The exact type of the keying material specified in the 'key' parameter MUST be defined in application profiles of this specification (REQ17), together with values of 'gkty' accepted by the application (REQ18). Additionally, documents specifying a type of keying material MUST register an entry in the "ACE Groupcomm Key Types" registry defined in Section 11.8, including its name, the corresponding value for the 'gkty parameter', and the application profile to be used with.

Name	Key Type Value	Profile	Description	Reference
Reserved	0		This value is reserved	[RFC-XXXX]

Figure 11: ACE Groupcomm Key Types

Note to RFC Editor: In Figure 11, please replace "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

The Join Response SHOULD contain the following parameters:

* 'exp', with value the expiration time of the keying material for the group communication, encoded as a CBOR unsigned integer. This field contains a numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds, analogous to what is specified for NumericDate in Section 2 of [RFC7519]. Group members MUST NOT use the keying material after the time indicated in this field, and they can retrieve the new group keying material from the KDC.

* 'exi', with value the residual lifetime of the keying material for the group communication, encoded as a CBOR unsigned integer. If the 'exp' parameter is included, this parameter MUST also be included. This field contains a numeric value representing the residual lifetime of the keying material in seconds, i.e., the number of seconds between the current time at the KDC and the time when the keying material expires (as specified in the 'exp' parameter, if present). A Client determines the expiration time of the keying material by adding the seconds specified in the 'exi' parameter to its current time upon receiving the response containing the 'exi' parameter. The Client MUST NOT use the keying material after such an expiration time, and it can retrieve the new group keying material from the KDC.

If a Client has a reliable way to synchronize its internal clock with UTC, and both the 'exp' and 'exi' parameters are present, then the Client MUST use the 'exp' parameter value as expiration time for the group keying material. Otherwise, the Client uses the 'exi' parameter value.

When a Client relies on the 'exi' parameter, the expiration time that it computes is offset in the future with respect to the actual expiration time as intended by the KDC and specified in the 'exp' parameter (if present). Such an offset is the amount of time between when the KDC sends the response message including the 'exi' parameter and when the Client receives that message. That is, especially if the delivery of the response to the Client is delayed, the Client will believe the keying material to be valid for a longer time than the KDC actually means. However, before approaching the actual expiration time, the KDC is expected to rekey the group and distribute new keying material (see Section 6).

Optionally, the Join Response MAY contain the following parameters, which, if included, MUST have the format and value as specified below.

* 'ace_groupcomm_profile', with value a CBOR integer that MUST be used to uniquely identify the application profile for group communication. Applications of this specification MUST register an application profile identifier and the related value for this parameter in the "ACE Groupcomm Profiles" registry (REQ19).

Name	Description	CBOR Value	Reference
Reserved	This value is reserved	0	[RFC-XXXX]

Figure 12: ACE Groupcomm Profiles

Note to RFC Editor: In Figure 12, please replace "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

* 'creds', which MUST be present if 'get_creds' was present in the request, otherwise it MUST NOT be present. This parameter is a CBOR array specifying the authentication credentials of the group members, i.e., of all of them or of the ones selected according to the 'get_creds' parameter in the request. In particular, each element of the array is a CBOR byte string, whose value is the original binary representation of a group member's authentication credential. It is REQUIRED of application profiles to define the specific formats of authentication credentials that are acceptable to use in the group (REQ6).

* 'peer_roles', which SHOULD be present if 'creds' is also present, otherwise it MUST NOT be present. This parameter is a CBOR array of n elements, where n is the number of authentication credentials included in the 'creds' parameter (at most the number of members in the group). The i-th element of the array specifies the role(s) that the group member associated with the i-th authentication credential in 'creds' has in the group. In particular, each array element is encoded like the role element of a scope entry, consistent with the used format (see Section 3.1).

  This parameter MAY be omitted if the Client can rely on other means to unambiguously gain knowledge of the role of each group member whose associated authentication credential is specified in the 'creds' parameter. For example, all such group members may have the same role in the group joined by the Client, and such a role can be unambiguously assumed by the Client (e.g., based on

what is defined in the used application profile of this specification). As another example, each of the authentication credentials specified in the 'creds' parameter can indicate the role(s) that the corresponding group member has in the group joined by the Client.

When receiving the authentication credential of a Client in the 'client_cred' parameter of a Join Request (see Section 4.3.1.1) or of an Authentication Credential Update Request (see Section 4.9.1.1), the KDC is not expected to check that the authentication credential indicates the role(s) that the Client can have or has in the group in question. When preparing a Join Response, the KDC can decide whether to include the 'peer_roles' parameter depending on the specific set of authentication credentials specified in the 'creds' parameter of that Join Response.

* 'peer_identifiers', which MUST be present if 'creds' is also present, otherwise it MUST NOT be present. This parameter is a CBOR array of n elements, where n is the number of authentication credentials included in the 'creds' parameter (at most the number of members in the group). The i-th element of the array specifies the node identifier that the group member associated with the i-th authentication credential in 'creds' has in the group. In particular, the i-th array element is encoded as a CBOR byte string, whose value is the node identifier of the group member. The specific format of node identifiers of group members is specified by the application profile (REQ25).

* 'group_policies', with value a CBOR map, whose entries specify how the group handles specific management aspects. These include, for instance, approaches to achieve synchronization of sequence numbers among group members. The elements of this field are registered in the "ACE Groupcomm Policies" registry. This specification defines the three elements "Sequence Number Synchronization Methods", "Key Update Check Interval", and "Expiration Delta", which are summarized in Figure 13. Application profiles that build on this document MUST specify the exact content format and default value of included map entries (REQ20).

Name	CBOR label	CBOR type	Description	Reference
Sequence Number Synchroniza-tion Method	0	tstr/int	Method for recipient group members to synchronize with sequence numbers of sender group members. Its value is taken from the 'Value' column of the Sequence Number Synchronization Method registry	[RFC-XXXX]
Key Update Check Interval	1	int	Polling interval in seconds, for group members to check at the KDC if the latest group keying material is the one that they store	[RFC-XXXX]
Expiration Delta	2	uint	Number of seconds from 'exp' until a UTC date/time, after which group members MUST stop using the group keying material that they store to decrypt incoming messages	[RFC-XXXX]

Figure 13: ACE Groupcomm Policies

Note to RFC Editor: In Figure 13, please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

* 'kdc_cred', encoded as a CBOR byte string, whose value is the original binary representation of the KDC's authentication credential. This parameter is used if the KDC has an associated authentication credential and this is required for the correct group operation. It is REQUIRED of application profiles to define whether the KDC has an authentication credential and if this has to be provided through the 'kdc_cred' parameter (REQ8).

In such a case, the KDC's authentication credential MUST have the same format used for the authentication credentials of the group members. It is REQUIRED of application profiles to define the specific formats that are acceptable to use for the authentication credentials in the group (REQ6).

* 'kdc_nonce', encoded as a CBOR byte string, and including a dedicated nonce N_KDC generated by the KDC. This parameter MUST be present if the 'kdc_cred' parameter is present.

* 'kdc_cred_verify', encoded as a CBOR byte string. This parameter MUST be present if the 'kdc_cred' parameter is present.

  This parameter contains a proof-of-possession (PoP) evidence computed by the KDC over the following PoP input: the nonce N_C (encoded as a CBOR byte string) concatenated with the nonce N_KDC (encoded as a CBOR byte string), where:

  - N_C is the nonce generated by the Client and specified in the 'cnonce' parameter of the Join Request, encoded as a CBOR byte string.

  - N_KDC is the nonce generated by the KDC and specified in the 'kdc_nonce' parameter, encoded as a CBOR byte string.

  An example of PoP input to compute 'kdc_cred_verify' using CBOR encoding is given in Figure 15.

  A possible type of PoP evidence is a signature that the KDC computes by using its own private key, whose corresponding public key is specified in the authentication credential carried in the 'kdc_cred' parameter. Application profiles of this specification MUST specify the exact approaches used by the KDC to compute the PoP evidence to include in 'kdc_cred_verify', and MUST specify which of those approaches is used in which case (REQ21).

* 'rekeying_scheme', identifying the rekeying scheme that the KDC uses to provide new group keying material to the group members. This parameter is encoded as a CBOR integer, whose value is taken from the "Value" column of the "ACE Groupcomm Rekeying Schemes" registry defined in Section 11.13 of this specification.

Value	Name	Description	Reference
0	Point-to-Point	The KDC individually targets each node to rekey, using the pairwise secure communication association with that node	[RFC-XXXX]

Figure 14: ACE Groupcomm Rekeying Schemes

Application profiles of this specification MAY define a default group rekeying scheme, to refer to in case the 'rekeying_scheme' parameter is not included in the Join Response (OPT9).

Note to RFC Editor: In Figure 14, please replace "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

* 'mgt_key_material', encoded as a CBOR byte string and containing the specific administrative keying material that the joining node requires in order to participate in the group rekeying process performed by the KDC. This parameter MUST NOT be present if the 'rekeying_scheme' parameter is not present and the application profile does not specify a default group rekeying scheme to use in the group. Some simple rekeying schemes may not require specific administrative keying material to be provided, e.g., the basic "Point-to-Point" group rekeying scheme (see Section 6.1).

  In more advanced group rekeying schemes, the administrative keying material can be composed of multiple keys organized, for instance, into a logical tree hierarchy, whose root key is the only administrative key shared by all the group members. In such a case, each group member is exclusively associated with one leaf key in the hierarchy, and stores only the administrative keys from the associated leaf key all the way up along the path to the root key. That is, different group members can be provided with a different subset of the overall administrative keying material.

  It is expected from separate documents to define how the advanced group rekeying scheme possibly indicated in the 'rekeying_scheme' parameter is used by an application profile of this specification. This includes defining the format of the administrative keying material to specify in 'mgt_key_material', consistently with the group rekeying scheme and the application profile in question.

* 'control_group_uri', with a full URI as value, encoded as a CBOR text string. The URI MUST specify addressing information intended to reach all the members in the group. For example, this can be a

multicast IP address, optionally together with a port number that, if omitted, defaults to 5683, i.e., the default port number for the "coap" URI scheme (see Section 6.1 of [RFC7252]). The URI MUST include GROUPNAME in the url-path. A default url-path is /ace-group/GROUPNAME, although implementations can use different ones instead. The URI MUST NOT have url-path /ace-group/GROUPNAME/node.

If 'control_group_uri' is included in the Join Response, the Clients supporting this parameter act as CoAP servers, host a resource at this specific URI, and listen to the specified addressing information.

The KDC MAY use this URI to send one-to-many CoAP requests to the Client group members (acting as CoAP servers in this exchange), for example for one-to-many provisioning of new group keying material when performing a group rekeying (see Section 4.8.1.1), or to inform the Clients of their removal from the group (see Section 5).

In particular, this resource is intended for communications concerning exclusively the group identified by GROUPNAME and whose group name was specified in the 'scope' parameter of the Join Request, if present. If the KDC does not implement mechanisms using this resource for that group, it can ignore this parameter. Other additional functionalities of this resource MAY be defined in application profiles of this specifications (OPT10).

```
N_C and N_KDC expressed in CBOR diagnostic notation:
  N_C   = h'25a8991cd700ac01'
  N_KDC = h'cef04b2aa791bc6d'


N_C and N_KDC as CBOR encoded byte strings:
  N_C   = 0x4825a8991cd700ac01
  N_KDC = 0x48cef04b2aa791bc6d

PoP input:
  0x48 25a8991cd700ac01 48 cef04b2aa791bc6d
```

Figure 15: Example of PoP input to compute 'kdc_cred_verify' using CBOR encoding

After sending the Join Response, if the KDC has an associated authentication credential, the KDC MUST store the N_C value specified in the 'cnonce' parameter of the Join Request, as a 'clientchallenge' value associated with the Client, replacing the currently stored value (if any). If, as a group member, the Client later sends a GET

request to the /ace-group/GROUPNAME/kdc-cred resource for retrieving the latest KDC's authentication credential (see Section 4.5.1), then the KDC is able to use the stored 'clientchallenge' for computing a PoP evidence to include in the response sent to the Client, hence proving the possession of its own private key.

If the Join Response includes the 'kdc_cred_verify' parameter, the Client verifies the conveyed PoP evidence and considers the group joining unsuccessful in case of failed verification. Application profiles of this specification MUST specify the exact approaches used by the Client to verify the PoP evidence in 'kdc_cred_verify', and MUST specify which of those approaches is used in which case (REQ21).

Specific application profiles that build on this document MUST specify the communication protocol that members of the group use to communicate with each other (REQ22) and how exactly the keying material is used to protect the group communication (REQ23).

4.3.1.1. Join the Group

Figure 16 gives an overview of the join exchange between the Client and the KDC, when the Client first joins a group, while Figure 17 shows an example.

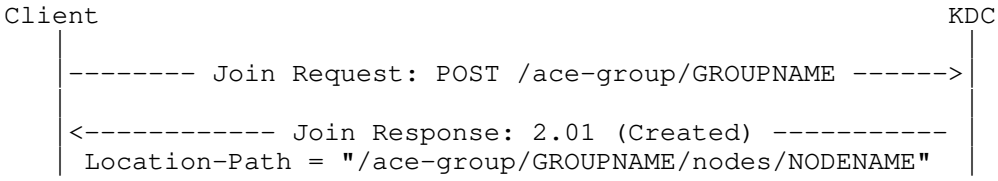


Figure 16: Message Flow of the Join Request-Response

```
Request:

Header: POST (Code=0.02)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation,
         with AUTH_CRED and POP_EVIDENCE being CBOR byte strings):
  { "scope": << [ "group1", ["sender", "receiver"] ] >> ,
    "get_creds": [true, ["sender"], []], "client_cred": AUTH_CRED,
    "cnonce": h'25a8991cd700ac01', "client_cred_verify": POP_EVIDENCE }

Response:

Header: Created (Code=2.01)
Content-Format: "application/ace-groupcomm+cbor"
Location-Path: "kdc.example.com"
Location-Path: "g1"
Location-Path: "nodes"
Location-Path: "c101"
Payload (in CBOR diagnostic notation,
         with KEY, AUTH_CRED_1, AUTH_CRED_2,
         ID_1, and ID_2 being CBOR byte strings):
  { "gkty": 13, "key": KEY, "num": 12, "exp": 1924992000,
    "exi": 2592000, "creds": [ AUTH_CRED_1, AUTH_CRED_2 ],
    "peer_roles": ["sender", ["sender", "receiver"]],
    "peer_identifiers": [ ID1, ID2 ] }
```

Figure 17: Example of First Join Request-Response for Group Joining

If not previously established, the Client and the KDC MUST first establish a pairwise secure communication association (REQ24). This can be achieved, for instance, by using a transport profile of ACE. The join exchange MUST occur over that secure communication association. The Client and the KDC MAY use that same secure communication association to protect further pairwise communications that must be protected.

It is REQUIRED that the secure communication association between the Client and the KDC is established by using the proof-of-possession key bound to the access token. As a result, the proof-of-possession to bind the access token to the Client is performed by using the proof-of-possession key bound to the access token for establishing secure communication between the Client and the KDC.

To join the group, the Client sends a CoAP POST request to the /ace-group/GROUPNAME endpoint at the KDC, where the group to join is identified by GROUPNAME. The group name is specified in the scope entry conveyed by the 'scope' parameter of the request (if present), formatted as specified in Section 4.3.1. This group name is the same as in the scope entry corresponding to that group, specified in the 'scope' parameter of the Authorization Request/Response, or it can be retrieved from it. Note that, in case of successful joining, the Client will receive the URI to retrieve individual keying material and to leave the group in the Location-Path option of the response.

If the node is joining a group for the first time and the KDC maintains the authentication credentials of the group members, the Client is REQUIRED to send its own authentication credential and proof-of-possession (PoP) evidence in the Join Request (see the 'client_cred' and 'client_cred_verify' parameters in Section 4.3.1). The request is accepted only if both the authentication credential is provided and the PoP evidence is successfully verified.

If a node re-joins a group as authorized by the same access token and using the same authentication credential, it can omit the authentication credential and the PoP evidence, or just the PoP evidence, from the Join Request. Then, the KDC will be able to retrieve the node's authentication credential associated with the access token for that group. If the authentication credential has been discarded, the KDC replies with a 4.00 (Bad Request) error response, as specified in Section 4.3.1. If a node re-joins a group but wants to update its own authentication credential, it needs to include both its authentication credential and the PoP evidence in the Join Request like when it joined the group for the first time.

4.3.2. GET Handler

The GET handler returns the symmetric group keying material for the group identified by GROUPNAME.

The handler expects a GET request.

In addition to what is defined in Section 4.1.2, the handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler replies with a 2.05 (Content) response containing the symmetric group keying material. The payload of the response is formatted as a CBOR map which MUST contain the parameters 'gkty', 'key', and 'num' specified in Section 4.3.1.

Each of the following parameters specified in Section 4.3.1 MUST also be included in the payload of the response, if they are included in the payload of the Join Responses sent for the group: 'rekeying_scheme', 'mgt_key_material'.

The payload MAY also include the parameters 'ace_groupcomm_profile', 'exp', and 'exi' specified in Section 4.3.1. If the 'exp' parameter is included, the 'exi' parameter MUST also be included. If the parameter 'exi' is included, its value specifies the residual lifetime of the group keying material from the current time at the KDC.

4.3.2.1. Retrieve Group Keying Material

A node in the group can contact the KDC to retrieve the current group keying material, by sending a CoAP GET request to the /ace-group/GROUPNAME endpoint at the KDC, where the group is identified by GROUPNAME.

Figure 18 gives an overview of the key distribution exchange between the Client and the KDC, when the Client first joins a group, while Figure 19 shows an example.

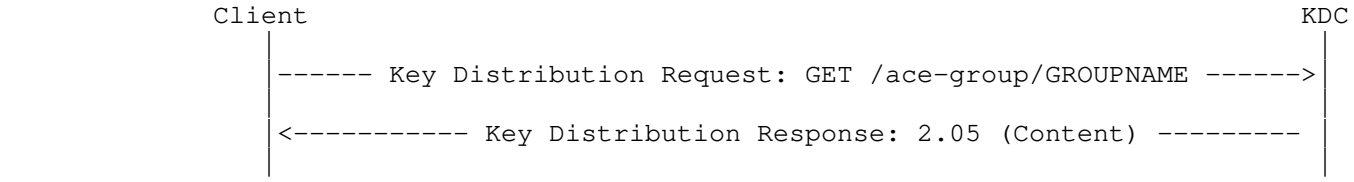


Figure 18: Message Flow of Key Distribution Request-Response

```
Request:

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Payload: -

Response:

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation,
         with KEY being a CBOR byte strings):
  { "gkty": 13, "key": KEY, "num": 12 }
```

Figure 19: Example of Key Distribution Request-Response

4.4. /ace-group/GROUPNAME/creds

This resource implements the GET and FETCH handlers.

4.4.1. FETCH Handler

The FETCH handler receives identifiers of group members for the group identified by GROUPNAME and returns the authentication credentials of such group members.

The handler expects a request with payload formatted as a CBOR map, which MUST contain the following field.

* 'get_creds', whose value is encoded as in Section 4.3.1 with the following modifications.

  - The arrays 'role_filter' and 'id_filter' MUST NOT both be empty, i.e., in CDDL notation: [bool, [ ], [ ]]. If the 'get_creds' parameter has such a format, the request MUST be considered malformed, and the KDC MUST reply with a 4.00 (Bad Request) error response.

    Note that a group member can retrieve the authentication credentials of all the current group members by sending a GET request to the same KDC resource instead (see Section 4.4.2.1).

  - The element 'inclusion_flag' encodes the CBOR simple value "true" (0xf5) or "false" (0xf4), as defined in Section 4.3.1.

- The array 'role_filter' can be empty, if the Client does not wish to filter the requested authentication credentials based on the roles of the group members.

- The array 'id_filter' contains zero or more node identifiers of group members, for the group identified by GROUPNAME, as defined in Section 4.3.1. The array may be empty, if the Client does not wish to filter the requested authentication credentials based on the node identifiers of the group members.

Note that, in case the 'role_filter' array and the 'id_filter' array are both non-empty:

* If the 'inclusion_flag' encodes the CBOR simple value "true" (0xf5), the handler returns the authentication credentials of group members whose roles match with 'role_filter' and/or having their node identifier specified in 'id_filter'.

* If the 'inclusion_flag' encodes the CBOR simple value "false" (0xf4), the handler returns the authentication credentials of group members whose roles match with 'role_filter' and, at the same time, not having their node identifier specified in 'id_filter'.

The specific format of authentication credentials as well as identifiers, roles, and combination of roles of group members MUST be specified by application profiles of this specification (REQ1, REQ6, REQ25).

The handler identifies the authentication credentials of the current group members for which either of the following holds:

* the role identifier matches with one of those indicated in the request; note that the request can specify a combination of roles, in which case the handler selects only the group members that have all the roles included in the combination.

* the node identifier matches with one of those indicated in the request, or does not match with any of those, consistent with the value of the element 'inclusion_flag'.

If all verifications succeed, the handler returns a 2.05 (Content) message response with payload formatted as a CBOR map, containing only the following parameters from Section 4.3.1.

* 'num', which encodes the version number of the current group keying material.

* 'creds', which encodes the list of authentication credentials of the selected group members.

* 'peer_roles', which encodes the role(s) that each of the selected group members has in the group.

  This parameter SHOULD be present and it MAY be omitted, according to the same criteria defined for the Join Response (see Section 4.3.1).

* 'peer_identifiers', which encodes the node identifier that each of the selected group members has in the group.

The specific format of authentication credentials as well as of node identifiers of group members is specified by the application profile (REQ6, REQ25).

If the KDC does not store any authentication credential associated with the specified node identifiers, the handler returns a response with payload formatted as a CBOR byte string of zero length.

The handler MAY enforce one of the following policies, in order to handle possible node identifiers that are included in the 'id_filter' element of the 'get_creds' parameter of the request but are not associated with any current group member. Such a policy MUST be specified by the application profile (REQ26).

* The KDC silently ignores those node identifiers.

* The KDC retains authentication credentials of group members for a given amount of time after their leaving, before discarding them. As long as such authentication credentials are retained, the KDC provides them to a requesting Client.

  If the KDC adopts this policy, the application profile MUST also specify the amount of time during which the KDC retains the authentication credential of a former group member after its leaving, possibly on a per-member basis.

Note that this resource handler only verifies that the node is authorized by the AS to access this resource. Nodes that are not members of the group but are authorized to do signature verifications on the group messages may be allowed to access this resource, if the application needs it.

4.4.1.1. Retrieve a Subset of Authentication Credentials in the Group

In case the KDC maintains the authentication credentials of group members, a node in the group can contact the KDC to request the authentication credentials, roles, and node identifiers of a specified subset of group members, by sending a CoAP FETCH request to the /ace-group/GROUPNAME/creds endpoint at the KDC, where the group is identified by GROUPNAME, and formatted as defined in Section 4.4.1.

Figure 20 gives an overview of the exchange mentioned above, while Figure 21 shows an example of such an exchange.

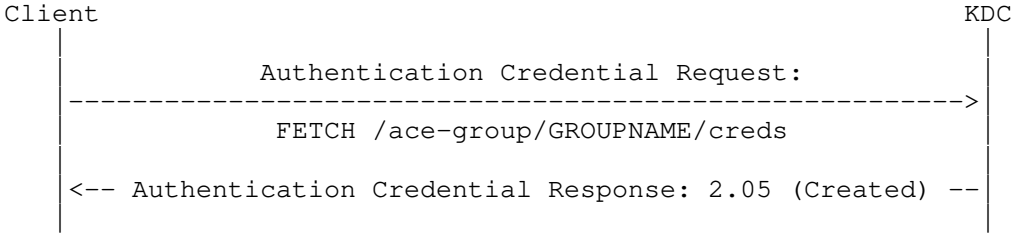


Figure 20: Message Flow of Authentication Credential Request-Response to Obtain the Authentication Credentials of Specific Group Members

Request:

```
Header: FETCH (Code=0.05)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "creds"
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
  { "get_creds": [true, [], [ ID_2, ID_3 ]] }
```

Response:

```
Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation,
         with AUTH_CRED_2, AUTH_CRED_3,
         ID_2, and ID_3 being CBOR byte strings):
  { "creds": [ AUTH_CRED_2, AUTH_CRED_3, ],
    "peer_roles": [ ["sender", "receiver"], "receiver" ],
    "peer_identifiers": [ ID_2, ID_3 ] }
```

Figure 21: Example of Authentication Credential Request-Response to Obtain the Authentication Credentials of Specific Group Members

4.4.2. GET Handler

The handler expects a GET request.

If all verifications succeed, the KDC replies with a 2.05 (Content) response as in the FETCH handler in Section 4.4.1, but specifying in the payload the authentication credentials of all the group members, together with their roles and node identifiers.

The parameter 'peer_roles' SHOULD be present in the payload of the response and it MAY be omitted, according to the same criteria defined for the Join Response (see Section 4.3.1).

4.4.2.1. Retrieve All Authentication Credentials in the Group

In case the KDC maintains the authentication credentials of group members, a group or an external signature verifier can contact the KDC to request the authentication credentials, roles, and node identifiers of all the current group members, by sending a CoAP GET request to the /ace-group/GROUPNAME/creds endpoint at the KDC, where the group is identified by GROUPNAME.

Figure 22 gives an overview of the message exchange, while Figure 23 shows an example of such an exchange.

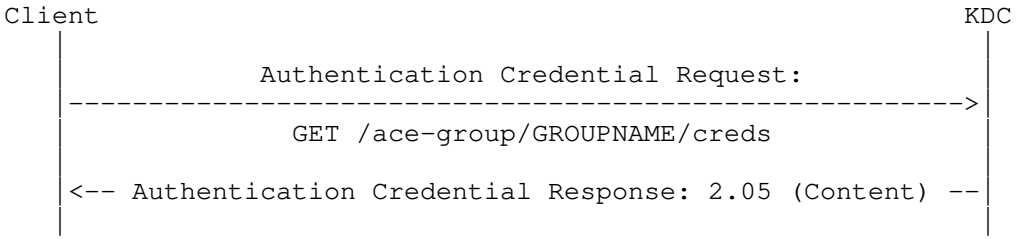


Figure 22: Message Flow of Authentication Credential Request-Response to Obtain the Authentication Credentials of all the Group Members

```
Request:

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "creds"
Payload: -

Response:

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation,
         with AUTH_CRED_1, AUTH_CRED_2, AUTH_CRED_3,
         ID_1, ID_2, and ID_3 being CBOR byte strings):
  { "num": 5,
    "creds": [ AUTH_CRED_1, AUTH_CRED_2, AUTH_CRED_3 ],
    "peer_roles": ["sender", ["sender", "receiver"], "receiver"],
    "peer_identifiers": [ ID_1, ID_2, ID_3 ] }
```

Figure 23: Example of Authentication Credential Request-Response to Obtain the Authentication Credentials of all the Group Members

4.5. /ace-group/GROUPNAME/kdc-cred

This resource implements a GET handler.

4.5.1. GET Handler

The handler expects a GET request.

If all verifications succeed, the handler returns a 2.05 (Content) message containing the KDC's authentication credential together with a proof-of-possession (PoP) evidence. The response MUST have Content-Format set to application/ace-groupcomm+cbor. The payload of the response is a CBOR map, which includes the following fields.

* The 'kdc_cred' parameter, specifying the KDC's authentication credential. This parameter is encoded like the 'kdc_cred' parameter in the Join Response (see Section 4.3.1).

* The 'kdc_nonce' parameter, specifying a nonce generated by the KDC. This parameter is encoded like the 'kdc_nonce' parameter in the Join Response (see Section 4.3.1).

* The 'kdc_cred_verify' parameter, specifying a PoP evidence computed by the KDC over the following PoP input: the nonce N_C (encoded as a CBOR byte string) concatenated with the nonce N_KDC (encoded as a CBOR byte string), where:

  - N_C is the nonce generated by the Client group member such that: i) the nonce was specified in the 'cnonce' parameter of the latest Join Request that the Client sent to the KDC in order to join the group identified by GROUPNAME; and ii) the KDC stored the nonce as 'clientchallenge' value associated with this Client as group member after sending the corresponding Join Response (see Section 4.3.1). This nonce is encoded as a CBOR byte string.

  - N_KDC is the nonce generated by the KDC and specified in the 'kdc_nonce' parameter, encoded as a CBOR byte string.

  An example of PoP input to compute 'kdc_cred_verify' using CBOR encoding is given in Figure 24.

  The PoP evidence is computed by means of the same method used for computing the PoP evidence that was included in the Join Response for this Client (see Section 4.3.1).

  Application profiles of this specification MUST specify the exact approaches used by the KDC to compute the PoP evidence to include in 'kdc_cred_verify', and MUST specify which of those approaches is used in which case (REQ21).

  If an application profile supports the presence of external signature verifiers that send GET requests to this resource, then the application profile MUST specify how external signature verifiers provide the KDC with a self-generated nonce to use as N_C (REQ21).

```
N_C and N_KDC expressed in CBOR diagnostic notation:
  N_C   = h'25a8991cd700ac01'
  N_KDC = h'0b7db12aaff56da3'


N_C and N_KDC as CBOR encoded byte strings:
  N_C   = 0x4825a8991cd700ac01
  N_KDC = 0x480b7db12aaff56da3

PoP input:
  0x48 25a8991cd700ac01 48 0b7db12aaff56da3
```

Figure 24: Example of PoP input to compute 'kdc_cred_verify' using CBOR encoding

4.5.1.1. Retrieve the KDC's Authentication Credential

In case the KDC has an associated authentication credential as required for the correct group operation, a group member or an external signature verifier can contact the KDC to request the KDC's authentication credential, by sending a CoAP GET request to the /ace-group/GROUPNAME/kdc-cred endpoint at the KDC, where GROUPNAME identifies the group.

Upon receiving the 2.05 (Content) response, the Client retrieves the KDC's authentication credential from the 'kdc_cred' parameter, and MUST verify the proof-of-possession (PoP) evidence specified in the 'kdc_cred_verify' parameter. In case of successful verification of the PoP evidence, the Client MUST store the obtained KDC's authentication credential and replace the currently stored one.

The PoP evidence is verified by means of the same method used when processing the Join Response (see Section 4.3.1). Application profiles of this specification MUST specify the exact approaches used by the Client to verify the PoP evidence in 'kdc_cred_verify', and MUST specify which of those approaches is used in which case (REQ21).

Figure 25 gives an overview of the exchange described above, while Figure 26 shows an example.

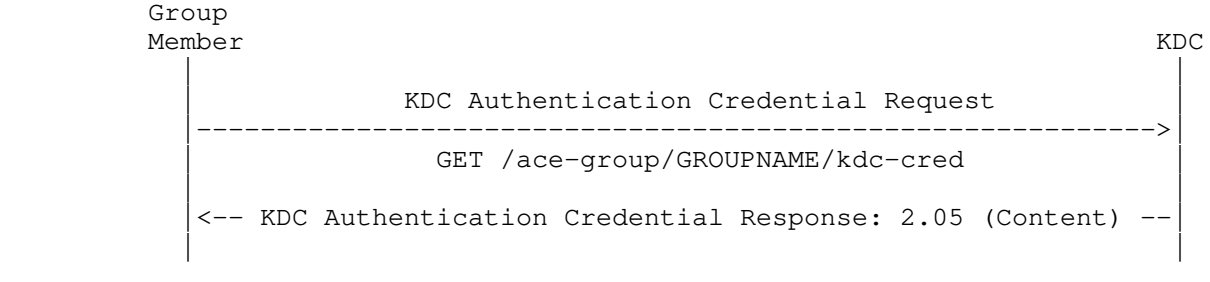


Figure 25: Message Flow of KDC Authentication Credential Request-Response to Obtain the Authentication Credential of the KDC

```
Request:

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "kdc-cred"
Payload: -

Response:

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation, with AUTH_CRED_KDC
         and POP_EVIDENCE being CBOR byte strings):
  {
    "kdc_nonce": h'0b7db12aaff56da3',
    "kdc_cred": AUTH_CRED_KDC,
    "kdc_cred_verify": POP_EVIDENCE
  }
```

Figure 26: Example of KDC Authentication Credential Request-Response to Obtain the Authentication Credential of the KDC

4.6. /ace-group/GROUPNAME/policies

This resource implements the GET handler.

4.6.1. GET Handler

The handler expects a GET request.

In addition to what is defined in Section 4.1.2, the handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler replies with a 2.05 (Content) response containing the list of policies for the group identified by GROUPNAME. The payload of the response is formatted as a CBOR map including only the parameter 'group_policies' defined in Section 4.3.1 and specifying the current policies in the group. If the KDC does not store any policy, the payload is formatted as a zero-length CBOR byte string.

The specific format and meaning of group policies MUST be specified in the application profile (REQ20).

4.6.1.1. Retrieve the Group Policies

A node in the group can contact the KDC to retrieve the current group policies, by sending a CoAP GET request to the /ace-group/GROUPNAME/policies endpoint at the KDC, where GROUPNAME identifies the group, and formatted as defined in Section 4.6.1

Figure 27 gives an overview of the exchange described above, while Figure 28 shows an example.

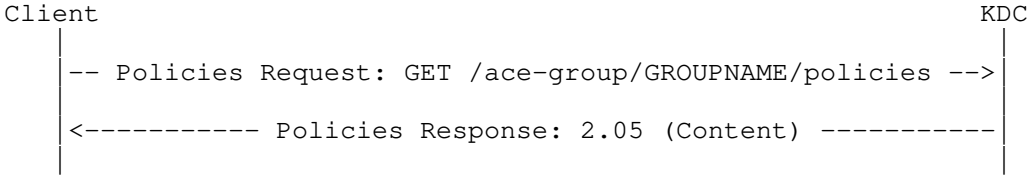


Figure 27: Message Flow of Policies Request-Response

Request:

```
Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "policies"
Payload: -
```

Response:

```
Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload(in CBOR diagnostic notation):
  { "group_policies": {"exp-delta": 120} }
```

Figure 28: Example of Policies Request-Response

4.7. /ace-group/GROUPNAME/num

This resource implements the GET handler.

4.7.1. GET Handler

The handler expects a GET request.

In addition to what is defined in Section 4.1.2, the handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler returns a 2.05 (Content) message containing an integer that represents the version number of the symmetric group keying material. This number is incremented on the KDC every time the KDC updates the symmetric group keying material, before the new keying material is distributed. This number is stored in persistent storage.

The payload of the response is formatted as a CBOR integer.

4.7.1.1. Retrieve the Keying Material Version

A node in the group can contact the KDC to request information about the version number of the symmetric group keying material, by sending a CoAP GET request to the /ace-group/GROUPNAME/num endpoint at the KDC, where GROUPNAME identifies the group, formatted as defined in Section 4.7.1. In particular, the version is incremented by the KDC every time the group keying material is renewed, before it is distributed to the group members.

Figure 29 gives an overview of the exchange described above, while Figure 30 shows an example.

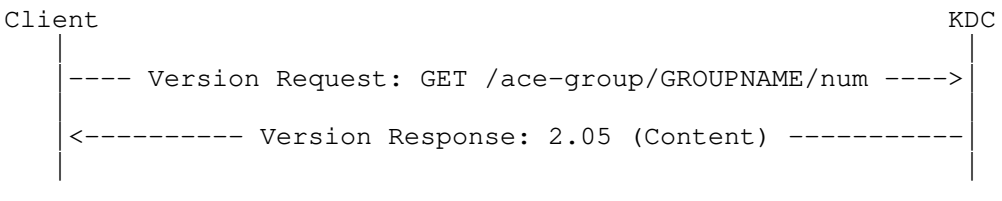


Figure 29: Message Flow of Version Request-Response

```
Request:

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "num"
Payload: -

Response:

Header: Content (Code=2.05)
Payload(in CBOR diagnostic notation):
  13
```

Figure 30: Example of Version Request-Response

4.8. /ace-group/GROUPNAME/nodes/NODENAME

This resource implements the GET, PUT, and DELETE handlers.

In addition to what is defined in Section 4.1.2, each of the handlers performs the following two verifications.

* The handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 0 ("Operation permitted only to group members").

* The handler verifies that the node name of the Client is equal to NODENAME used in the url-path. If the verification fails, the handler replies with a 4.03 (Forbidden) error response.

4.8.1. GET Handler

The handler expects a GET request.

If all verifications succeed, the handler replies with a 2.05 (Content) response containing both the group keying material and the individual keying material for the Client, or information enabling the Client to derive it.

The payload of the response is formatted as a CBOR map, which includes the same fields of the response defined in Section 4.3.2. In particular, the format for the group keying material is the same

as defined in the response of Section 4.3.2. If the 'exp' parameter is included, the 'exi' parameter MUST also be included. If the parameter 'exi' is included, its value specifies the residual lifetime of the group keying material from the current time at the KDC.

The CBOR map can include additional parameters that specify the individual keying material for the Client. The specific format of individual keying material for group members, or of the information to derive it, and corresponding CBOR label, MUST be specified in the application profile (REQ27) and registered in Section 11.7.

Optionally, the KDC can make the sub-resource at /ace-group/GROUPNAME/nodes/NODENAME also Observable [RFC7641] for the associated node. In case the KDC removes that node from the group without having been explicitly asked for it, this allows the KDC to send an unsolicited 4.04 (Not Found) response to the node as a notification of eviction from the group (see Section 5).

Note that the node could have also been observing the resource at /ace-group/GROUPNAME, in order to be informed of changes in the keying material. In such a case, this method would result in largely overlapping notifications received for the resource at /ace-group/GROUPNAME and the sub-resource at /ace-group/GROUPNAME/nodes/NODENAME.

In order to mitigate this, a node that supports the No-Response option [RFC7967] can use it when starting the observation of the sub-resource at /ace-group/GROUPNAME/nodes/NODENAME. In particular, the GET observation request can also include the No-Response option, with value set to 2 (Not interested in 2.xx responses).

4.8.1.1. Retrieve Group and Individual Keying Material

When any of the following happens, a node MUST stop using the stored group keying material to protect outgoing messages, and SHOULD stop using it to decrypt and verify incoming messages.

* Upon expiration of the keying material, according to what is indicated by the KDC with the 'exp' and/or 'exi' parameter (e.g., in a Join Response), or to a pre-configured value.

* Upon receiving a notification of revoked/renewed keying material from the KDC, possibly as part of an update of the keying material (rekeying) triggered by the KDC.

* Upon receiving messages from other group members without being able to retrieve the keying material to correctly decrypt them. This may be due to rekeying messages previously sent by the KDC, that the Client was not able to receive or decrypt.

In either case, if it wants to continue participating in the group communication, the Client has to request the latest keying material from the KDC. To this end, the Client sends a CoAP GET request to the /ace-group/GROUPNAME/nodes/NODENAME endpoint at the KDC, formatted as specified in Section 4.8.1. The Client can request the latest keying material from the KDC before the currently stored, old keying material reaches its expiration time.

Note that policies can be set up, so that the Client sends a Key Distribution Request to the KDC only after a given number of received messages could not be decrypted (because of failed decryption processing or inability to retrieve the necessary keying material).

It is application dependent and pertaining to the particular message exchange (e.g., [I-D.ietf-core-oscore-groupcomm]) to set up these policies for instructing Clients to retain incoming messages and for how long (OPT11). This allows Clients to possibly decrypt such messages after getting updated keying material, rather than just consider them invalid messages to discard right away.

After having failed to decrypt messages from another group member and having sent a Key Distribution Request to the KDC, the Client might end up retrieving the same, latest group keying material that it already stores. In such a case, multiple failed decryptions might be due to the message sender and/or the KDC that have changed their authentication credential. Hence, the Client can retrieve such latest authentication credentials, by sending to the KDC an Authentication Credential Request (see Section 4.4.1.1 and Section 4.4.2.1) and a KDC Authentication Credential Request (see Section 4.5.1.1), respectively.

The Client can also send to the KDC a Key Distribution Request without having been triggered by a failed decryption of a message from another group member, if the Client wants to be sure that it currently stores the latest group keying material. If that is the case, the Client will receive from the KDC the same group keying material it already stores.

Figure 31 gives an overview of the exchange described above, while Figure 32 shows an example.

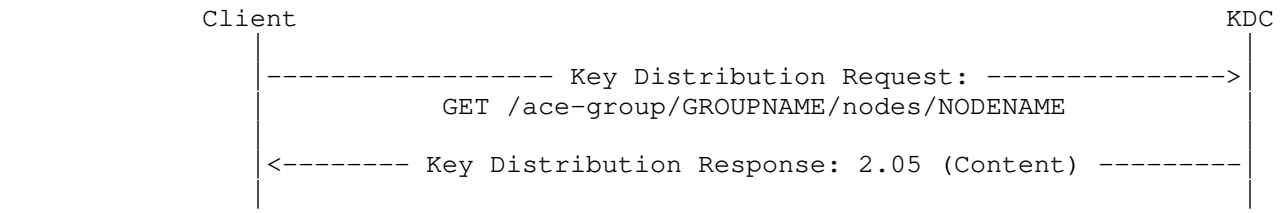


Figure 31: Message Flow of Key Distribution Request-Response

Request:

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "nodes"
Uri-Path: "c101"
Payload: -

Response:

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation,
         with KEY and IND_KEY being CBOR byte strings,
         and "ind-key" being the profile-specified label
         for individual keying material):
  { "gkty": 13, "key": KEY, "num": 12, "ind-key": IND_KEY }

Figure 32: Example of Key Distribution Request-Response

4.8.2. PUT Handler

The PUT handler processes requests from a Client that asks for new individual keying material, as required to process messages exchanged in the group.

The handler expects a PUT request with empty payload.

In addition to what is defined in Section 4.1.2 and at the beginning of Section 4.8, the handler verifies that this operation is consistent with the set of roles that the Client has in the group (REQ11). If the verification fails, the KDC MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 1 ("Request inconsistent with the current roles").

If the KDC is currently not able to serve this request, i.e., to generate new individual keying material for the requesting Client, the KDC MUST reply with a 5.03 (Service Unavailable) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 4 ("No available node identifiers").

If all verifications succeed, the handler replies with a 2.05 (Content) response containing newly generated, individual keying material for the Client. The payload of the response is formatted as a CBOR map. The specific format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label, MUST be specified in the application profile (REQ27) and registered in Section 11.7.

The typical successful outcome consists in replying with newly generated, individual keying material for the Client, as defined above. However, application profiles of this specification MAY also extend this handler in order to achieve different akin outcomes (OPT12), for instance:

* Not providing the Client with newly generated, individual keying material, but rather rekeying the whole group, i.e., providing all the current group members with newly generated group keying material.

* Both providing the Client with newly generated, individual keying material, as well as rekeying the whole group, i.e., providing all the current group members with newly generated group keying material.

In either case, the handler may specify the new group keying material as part of the 2.05 (Content) response.

Note that this handler is not intended to accommodate requests from a group member to trigger a group rekeying, whose scheduling and execution is an exclusive prerogative of the KDC (see also related security considerations in Section 10.2).

4.8.2.1. Request to Change Individual Keying Material

A Client may ask the KDC for new, individual keying material. For instance, this can be due to the expiration of such individual keying material, or to the exhaustion of AEAD nonces, if an AEAD encryption algorithm is used for protecting communications in the group. An example of individual keying material can simply be an individual encryption key associated with the Client. Hence, the Client may ask for a new individual encryption key, or for new input material to derive it.

To this end, the Client performs a Key Renewal Request-Response exchange with the KDC, i.e., it sends a CoAP PUT request to the /ace-group/GROUPNAME/nodes/NODENAME endpoint at the KDC, where GROUPNAME identifies the group and NODENAME is its node name, and formatted as defined in Section 4.8.1.

Figure 33 gives an overview of the exchange described above, while Figure 34 shows an example.

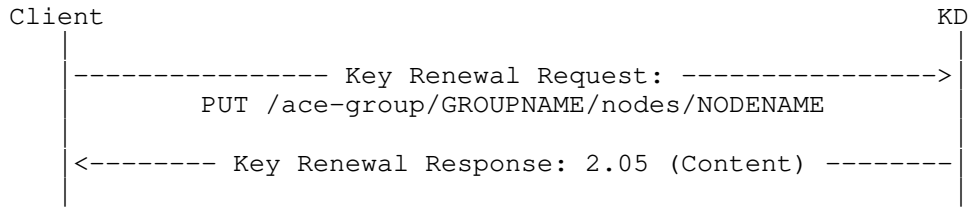


Figure 33: Message Flow of Key Renewal Request-Response

```
Request:

Header: PUT (Code=0.03)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "nodes"
Uri-Path: "c101"
Payload: -

Response:

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation, with IND_KEY being a
         CBOR byte string, and "ind-key" being the profile-specified
         label for individual keying material):
  { "ind-key": IND_KEY }
```

Figure 34: Example of Key Renewal Request-Response

Note that there is a difference between the Key Renewal Request in this section and the Key Distribution Request in Section 4.8.1.1. The former asks the KDC for new individual keying material, while the latter asks the KDC for the current group keying material together with the current individual keying material.

As discussed in Section 4.8.2, application profiles of this specification may define alternative outcomes for the Key Renewal Request-Response exchange (OPT12), where the provisioning of new individual keying material is replaced by or combined with the execution of a whole group rekeying.

4.8.3. DELETE Handler

The DELETE handler removes the node identified by NODENAME from the group identified by GROUPNAME.

The handler expects a DELETE request with empty payload.

In addition to what is defined in Section 4.1.2, the handler verifies that the Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 0 ("Operation permitted only to group members").

If all verification succeeds, the handler performs the actions defined in Section 5 and replies with a 2.02 (Deleted) response with empty payload.

4.8.3.1. Leave the Group

A Client can actively request to leave the group. In this case, the Client sends a CoAP DELETE request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME at the KDC, where GROUPNAME identifies the group and NODENAME is its node name, formatted as defined in Section 4.8.3

Note that, after having left the group, the Client may wish to join it again. Then, as long as the Client is still authorized to join the group, i.e., the associated access token is still valid, the Client can request to re-join the group directly to the KDC (see Section 4.3.1.1), without having to retrieve a new access token from the AS.

4.9. /ace-group/GROUPNAME/nodes/NODENAME/cred

This resource implements the POST handler.

4.9.1. POST Handler

The POST handler is used to replace the stored authentication credential of this Client (identified by NODENAME) with the one specified in the request at the KDC, for the group identified by GROUPNAME.

The handler expects a POST request with payload as specified in Section 4.3.1, with the difference that it includes only the parameters 'client_cred', 'cnonce', and 'client_cred_verify'.

The PoP evidence included in the 'client_cred_verify' parameter is computed in the same way considered in Section 4.3.1 and defined by the specific application profile (REQ14), by using the following to build the PoP input: i) the same scope entry specified by the Client in the 'scope' parameter of the latest Join Request that the Client sent to the KDC in order to join the group identified by GROUPNAME; ii) the latest N_S value stored by the Client; iii) a new N_C nonce generated by the Client and specified in the parameter 'cnonce' of this request.

An example of PoP input to compute 'client_cred_verify' using CBOR encoding is given in Figure 35.

It is REQUIRED of application profiles to define the specific formats of authentication credentials that are acceptable to use in the group (REQ6).

In addition to what is defined in Section 4.1.2 and at the beginning of Section 4.8, the handler verifies that this operation is consistent with the set of roles that the node has in the group. If the verification fails, the KDC MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 1 ("Request inconsistent with the current roles").

If the KDC cannot retrieve the 'kdcchallenge' associated with this Client (see Section 3.3), the KDC MUST reply with a 4.00 (Bad Request) error response, which MUST also have Content-Format application/ace-groupcomm+cbor. The payload of the error response is a CBOR map including a newly generated 'kdcchallenge' value. This is specified in the 'kdcchallenge' parameter. In such a case the KDC MUST store the newly generated value as the 'kdcchallenge' value associated with this Client, replacing the currently stored value (if any).

Otherwise, the handler checks that the authentication credential specified in the 'client_cred' field is valid for the group identified by GROUPNAME. That is, the handler checks that the authentication credential is encoded according to the format used in the group, is intended for the public key algorithm used in the group, and is aligned with the possible associated parameters used in the group. If that cannot be successfully verified, the handler MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 2 ("Authentication credential incompatible with the group configuration").

Otherwise, the handler verifies the PoP evidence contained in the 'client_cred_verify' field of the request, by using the authentication credential specified in the 'client_cred' field, as well as the same way considered in Section 4.3.1 and defined by the specific application profile (REQ14). If the PoP evidence does not pass verification, the handler MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 3 ("Invalid Proof-of-Possession evidence").

If all verifications succeed, the handler performs the following actions.

* The handler associates the authentication credential from the 'client_cred' field of the request with the node identifier NODENAME, as well as with the access token associated with the node identified by NODENAME.

* In the stored list of group members' authentication credentials for the group identified by GROUPNAME, the handler replaces the authentication credential of the node identified by NODENAME with the authentication credential specified in the 'client_cred' field of the request.

Then, the handler replies with a 2.04 (Changed) response, which does not include a payload.

```
scope, N_S, and N_C expressed in CBOR diagnostic notation:
  scope = h'826667726f7570316673656e646572'
  N_S   = h'018a278f7faab55a'
  N_C   = h'0446baefc56111bf'


scope, N_S, and N_C as CBOR encoded byte strings:
  scope = 0x4f826667726F7570316673656E646572
  N_S   = 0x48018a278f7faab55a
  N_C   = 0x480446baefc56111bf

PoP input:
  0x4f 826667726f7570316673656e646572
    48 018a278f7faab55a 48 0446baefc56111bf
```

Figure 35: Example of PoP input to compute 'client_cred_verify' using CBOR encoding

4.9.1.1. Uploading an Authentication Credential

In case the KDC maintains the authentication credentials of group members, a node in the group can contact the KDC to upload a new authentication credential to use in the group, and to replace the currently stored one.

To this end, the Client performs an Authentication Credential Update Request-Response exchange with the KDC, i.e., it sends a CoAP POST request to the /ace-group/GROUPNAME/nodes/NODENAME/cred endpoint at the KDC, where GROUPNAME identifies the group and NODENAME is its node name.

The request is formatted as specified in Section 4.9.1.

Figure 36 gives an overview of the exchange described above, while Figure 37 shows an example.

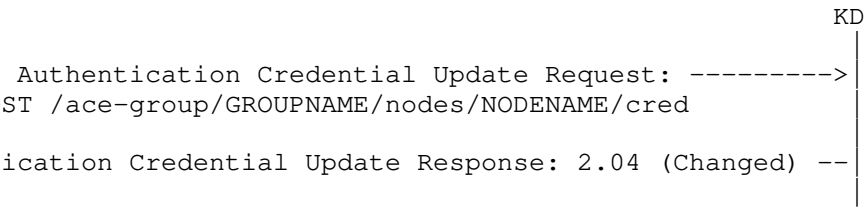


Figure 36: Message Flow of Authentication Credential Update Request-Response

Request:

```
Header: POST (Code=0.02)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "nodes"
Uri-Path: "c101"
Uri-Path: "cred"
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation, with AUTH_CRED
         and POP_EVIDENCE being CBOR byte strings):
  { "client_cred": AUTH_CRED, "cnonce": h'0446baefc56111bf',
    "client_cred_verify": POP_EVIDENCE }
```

Response:

```
Header: Changed (Code=2.04)
Payload: -
```

Figure 37: Example of Authentication Credential Update Request-Response

Additionally, after updating its own authentication credential, a group member MAY send to the group a number of requests including an identifier of the updated authentication credential, to notify other group members that they have to retrieve it. How this is done depends on the group communication protocol used, and therefore is application profile specific (OPT13).

5. Removal of a Group Member

A Client identified by NODENAME may be removed from a group identified by GROUPNAME where it is a member, for example due to the following reasons.

1. The Client explicitly asks to leave the group, as defined in Section 4.8.3.1.

2. The node has been found compromised or is suspected so. The KDC is expected to determine that a group member has to be evicted either through its own means, or based on information that it obtains from a trusted source (e.g., an Intrusion Detection System, or an issuer of authentication credentials). Additional mechanics, protocols, and interfaces at the KDC that can support this are out of the scope of this document.

3. The Client's authorization to be a group member with the current roles is not valid anymore, i.e., the access token has expired or has been revoked. If the AS provides token introspection (see Section 5.9 of [RFC9200]), the KDC can optionally use it and check whether the Client is still authorized.

In either case, the KDC performs the following actions.

* The KDC removes the Client from the list of current members of the group. When doing so, the KDC deletes the currently stored value of 'clientchallenge' for that Client, which was specified in the latest Join Request that the Client sent to the KDC in order to join the group (see Section 4.3.1).

* In case of forced eviction, i.e., for cases 2 and 3 above, the KDC deletes the authentication credential of the removed Client, if it acts as a repository of authentication credentials for group members.

* If the removed Client is registered as an observer of the group-membership resource at /ace-group/GROUPNAME, the KDC removes the Client from the list of observers of that resource.

* If the sub-resource nodes/NODENAME was created for the removed Client, the KDC deletes that sub-resource.

  In case of forced eviction, i.e., for cases 2 and 3 above, the KDC MAY explicitly inform the removed Client, by means of the following methods.

  - If the evicted Client implements the 'control_uri' resource specified in Section 4.3.1, the KDC sends a DELETE request, targeting the URI specified in the 'control_uri' parameter of the Join Request (see Section 4.3.1).

  - If the evicted Client is observing its associated sub-resource at /ace-group/GROUPNAME/nodes/NODENAME (see Section 4.8.1), the KDC sends an unsolicited 4.04 (Not Found) error response, which does not include the Observe option and indicates that the observed resource has been deleted (see Section 3.2 of [RFC7641]).

    The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 5 ("Group membership terminated").

* If forward security is prescribed by application policies installed at the KDC or by the used application profile of this specification, then the KDC MUST generate new group keying material and securely distribute it to all the current group members except the leaving node (see Section 6).

6. Group Rekeying Process

A group rekeying is started and driven by the KDC. The KDC is not intended to accommodate explicit requests from group members to trigger a group rekeying. That is, the scheduling and execution of a group rekeying is an exclusive prerogative of the KDC. Reasons that can trigger a group rekeying are a change in the group membership, the current group keying material approaching its expiration time, or a regularly scheduled update of the group keying material.

The KDC can perform a group rekeying before the current group keying material expires, unless it is acceptable or there are reasons to temporarily pause secure communications in the group, following the

expiration of the current keying material. For example, a pause in the group communication might have been scheduled to start anyway when the group keying material expires, e.g., to allow maintenance operations on the group members. As another example, the KDC might be carrying out a verification that some group members are seemingly compromised and to be evicted, and this requires to be completed in order to appropriately define and schedule the exact rekeying process to perform. As a result, the KDC could delay the execution of the group rekeying.

The KDC MUST increment the version number NUM of the current keying material, before distributing the newly generated keying material with version number NUM+1 to the group. Once the group rekeying is completed, the KDC MUST delete the old keying material and SHOULD store the newly distributed keying material in persistent storage.

Distributing the new group keying material requires the KDC to send multiple rekeying messages to the group members. Depending on the rekeying scheme used in the group and the reason that has triggered the rekeying process, each rekeying message can be intended for one or multiple group members, hereafter referred to as target group members. The KDC MUST support at least the "Point-to-Point" group rekeying scheme in Section 6.1 and MAY support additional ones.

Each rekeying message MUST have Content-Format set to application/ace-groupcomm+cbor and its payload formatted as a CBOR map, which MUST include at least the information specified in the Key Distribution Response message (see Section 4.3.2), i.e., the parameters 'gkty', 'key', and 'num' defined in Section 4.3.1. The CBOR map SHOULD also include the parameters 'exp' and 'exi'. If the 'exp' parameter is included, the 'exi' parameter MUST also be included. The CBOR map MAY include the parameter 'mgt_key_material' specifying new administrative keying material for the target group members, if relevant for the used rekeying scheme.

A rekeying message may include additional information, depending on the rekeying scheme used in the group, the reason that has triggered the rekeying process, and the specific target group members. In particular, if the group rekeying is performed due to one or multiple Clients that have joined the group and the KDC acts as a repository of authentication credentials of the group members, then a rekeying message MAY also include the authentication credentials that those Clients use in the group, together with the roles and node identifier that the corresponding Client has in the group. It is RECOMMENDED to specify this information by means of the parameters 'creds', 'peer_roles', and 'peer_identifiers', like it is done in the Join Response message (see Section 4.3.1).

The complete format of a rekeying message, including the encoding and content of the 'mgt_key_material' parameter, has to be defined in separate specifications aimed at profiling the used rekeying scheme in the context of the used application profile of this specification. As a particular case, an application profile of this specification MAY define additional information to include in rekeying messages for the "Point-to-Point" group rekeying scheme in Section 6.1 (OPT14).

Consistently with the used group rekeying scheme, the actual delivery of rekeying messages can occur through different approaches, as discussed in the following Section 6.1 and Section 6.2.

The possible, temporary misalignment of the keying material stored by the different group members due to a group rekeying is discussed in Section 6.3. Further security considerations related to the group rekeying process are compiled in Section 10.2.

6.1. Point-to-Point Group Rekeying

A point-to-point group rekeying consists in the KDC sending one individual rekeying message to each target group member. In particular, the rekeying message is protected by means of the security association between the KDC and the target group member in question, as per the used application profile of this specification and the used transport profile of ACE.

This is the approach taken by the basic "Point-to-Point" group rekeying scheme, that the KDC can explicitly signal in the Join Response (see Section 4.3.1), through the 'rekeying_scheme' parameter specifying the value 0.

When taking this approach in the group identified by GROUPNAME, the KDC can practically deliver the rekeying messages to the target group members in different, co-existing ways.

* The KDC SHOULD make the /ace-group/GROUPNAME resource Observable [RFC7641]. Thus, upon performing a group rekeying, the KDC can distribute the new group keying material through individual notification responses sent to the target group members that are also observing that resource.

In case the KDC deletes the group (and thus deletes the /ace-group/GROUPNAME resource), relying on CoAP Observe as discussed above also allows the KDC to send an unsolicited 4.04 (Not Found) response to each observer group member, as a notification of group termination. The response MUST have Content-Format set to application/concise-problem-details+cbor and is formatted as defined in Section 4.1.2. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 6 ("Group deleted").

* If a target group member specified a URI in the 'control_uri' parameter of the Join Request upon joining the group (see Section 4.3.1), the KDC can provide that group member with the new group keying material by sending a unicast POST request to that URI.

  A Client that does not plan to observe the /ace-group/GROUPNAME resource at the KDC SHOULD provide a URI in the 'control_uri' parameter of the Join Request upon joining the group.

If the KDC has to send a rekeying message to a target group member, but this did not include the 'control_uri' parameter in the Join Request and is not a registered observer for the /ace-group/GROUPNAME resource, then that target group member would not be able to participate in the group rekeying. Later on, after having repeatedly failed to successfully exchange secure messages in the group, that group member can retrieve the current group keying material from the KDC, by sending a GET request to /ace-group/GROUPNAME or /ace-group/GROUPNAME/nodes/NODENAME (see Section 4.3.2 and Section 4.8.1, respectively).

Figure 38 provides an example of point-to-point group rekeying. In particular, the example makes the following assumptions.

* The group currently consists of four group members, namely C1, C2, C3, and C4.

* Each group member, when joining the group, provided the KDC with a URI in the 'control_uri' parameter, with url-path "grp-rek".

* Before the group rekeying is performed, the keying material used in the group has version number num=5.

* The KDC performs the group rekeying in such a way to evict the group member C3, which has been found to be compromised.

In the example, the KDC individually rekeys the group members intended to remain in the group (i.e., C1, C2, and C4), by means of one rekeying message each.

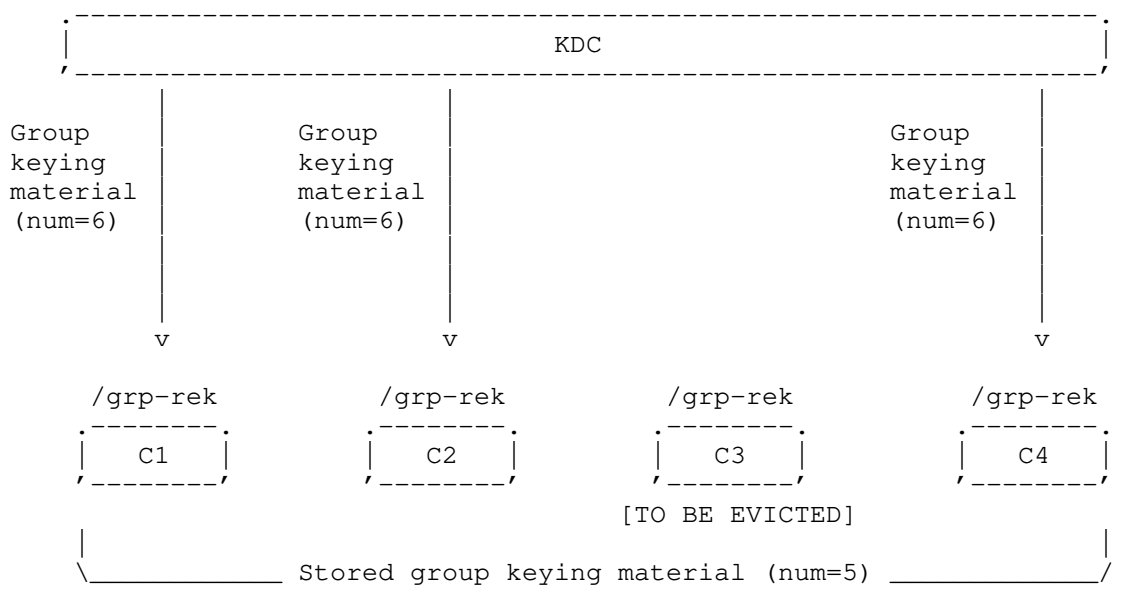


Figure 38: Example of Message Exchanges for a Point-to-Point Group Rekeying

6.2. One-to-Many Group Rekeying

This section provides high-level recommendations on how the KDC can rekey a group by means of a more efficient and scalable group rekeying scheme, e.g., [RFC2093][RFC2094][RFC2627]. That is, each rekeying message might be, and likely is, intended for multiple target group members, and thus can be delivered to the whole group, although possible to decrypt only for the actual target group members.

This yields an overall lower number of rekeying messages, thus potentially reducing the overall time required to rekey the group. On the other hand, it requires the KDC to provide and use additional administrative keying material to protect the rekeying messages, and to additionally sign them to ensure source authentication (see Section 6.2.1).

Compared to a group rekeying performed in a point-to-point fashion (see Section 6.1), a one-to-many group rekeying typically pays off in large-scale groups, due to the reduced time for completing the

rekeying, a more efficient utilization of network resources, and a reduced performance overhead at the KDC. To different extents, it also requires individual group members to locally perform additional operations, in order to handle the administrative keying material and verify source authentication of rekeying messages. Therefore, one-to-many group rekeying schemes and their employment ought to ensure that the experienced performance overhead on the group members remains bearable also for resource-constrained devices.

The exact set of rekeying messages to send, their content and format, the administrative keying material to use to protect them, as well as the set of target group members depend on the specific group rekeying scheme, and are typically affected by the reason that has triggered the group rekeying. Details about the data content and format of rekeying messages have to be defined by separate documents profiling the use of the group rekeying scheme, in the context of the used application profile of this specification.

When one of these group rekeying schemes is used, the KDC provides a number of related information to a Client joining the group in the Join Response message (see Section 4.3.1). In particular, 'rekeying_scheme' identifies the rekeying scheme used in the group (if no default can be assumed); 'control_group_uri', if present, specifies a URI whose addressing information is, e.g., a multicast IP address, and where the KDC will send the rekeying messages for that group by reaching all the group members; 'mgt_key_material' specifies a subset of the administrative keying material intended for that particular joining Client to have, as used to protect the rekeying messages sent to the group when intended also to that joining Client.

Rekeying messages can be protected at the application layer, by using COSE and the administrative keying material as prescribed by the specific group rekeying scheme (see Section 6.2.1). After that, the delivery of protected rekeying messages to the intended target group members can occur in different ways, such as the following ones.

* Over multicast - In this case, the KDC simply sends a rekeying message as a CoAP request addressed to the URI specified in the 'control_group_uri' parameter of the Join Response (see Section 4.3.1).

  If a particular rekeying message is intended for a single target group member, the KDC may alternatively protect the message using the security association with that group member, and deliver the message like when using the "Point-to-Point" group rekeying scheme (see Section 6.1).

* Through a pub-sub communication model - In this case, the KDC acts as a publisher and publishes each rekeying message to a specific "rekeying topic", which is associated with the group and is hosted at a broker server. Following their group joining, the group members subscribe to the rekeying topic at the broker, thus receiving the group rekeying messages as they are published by the KDC.

  In order to make such message delivery more efficient, the rekeying topic associated with a group can be further organized into subtopics. For instance, the KDC can use a particular subtopic to address a particular set of target group members during the rekeying process, as possibly aligned to a similar organization of the administrative keying material (e.g., a key hierarchy).

  The setup of rekeying topics at the broker as well as the discovery of the topics at the broker for group members are application specific. A possible way is for the KDC to provide such information in the Join Response message (see Section 4.3.1), by means of a new parameter analogous to 'control_group_uri' and specifying the URI(s) of the rekeying topic(s) that a group member has to subscribe to at the broker.

Regardless of the specifically used delivery method, the group rekeying scheme can perform a possible roll-over of the administrative keying material through the same sent rekeying messages. Actually, such a roll-over occurs every time a group rekeying is performed upon the leaving of group members, which have to be excluded from future communications in the group.

From a high level point of view, each group member stores only a subset of the overall administrative keying material, obtained upon joining the group. Then, when a group rekeying occurs:

* Each rekeying message is protected by using a (most convenient) key from the administrative keying material such that: i) the used key is not stored by any node leaving the group, i.e., the key is safe to use and does not have to be renewed; and ii) the used key is stored by all the target group members, that indeed have to be provided with new group keying material to protect communications in the group.

* Each rekeying message includes not only the new group keying material intended for all the rekeyed group members, but also any new administrative keys that: i) are pertaining to and supposed to be stored by the target group members; and ii) had to be updated since leaving group members store the previous version.

Further details depend on the specific rekeying scheme used in the group.

Figure 39 provides an example of one-to-many group rekeying over multicast. In particular, the example makes the following assumptions.

* The group currently consists of four group members, namely C1, C2, C3, and C4.

* Each group member, when joining the group, provided the KDC with a URI in the 'control_uri' parameter, with url-path "grp-rek".

* Each group member, when joining the group, received from the KDC a URI in the 'control_group_uri' parameter, specifying the multicast address MULT_ADDR and url-path "grp-mrek".

* Before the group rekeying is performed, the keying material used in the group has version number num=5.

* The KDC performs the group rekeying in such a way to evict the group member C3, which has been found to be compromised.

In the example, the KDC determines that the most convenient way to perform a group rekeying that evicts C3 is as follows.

First, the KDC sends one rekeying message over multicast, to the multicast address MULT_ADDR and the url-path "grp-mrek". In the figure, the message is denoted with dashed lines. The message is protected with a non-compromised key from the administrative keying material that only C1 and C2 store. Therefore, even though all the group members receive this message, only C1 and C2 are able to decrypt it. The message includes: the new group keying material with version number num=6; and new keys from the administrative keying material to replace those stored by the group members C1, C2, and C3.

After that, the KDC sends one rekeying message addressed individually to C4 and with url-path "grp-rek". In the figure, the message is denoted with a dotted line. The message is protected with the secure association shared between C4 and the KDC. The message includes: the new group keying material with version number num=6; and new keys from the administrative keying material to replace those stored by both C4 and C3.

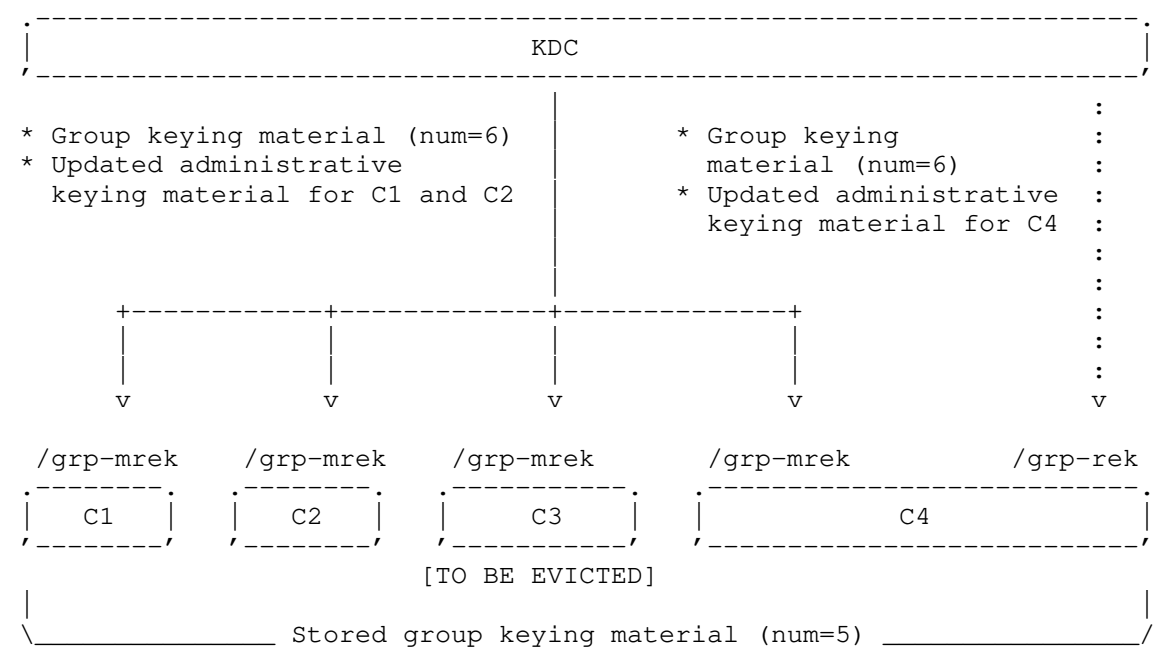


Figure 39: Example of Message Exchanges for a One-to-Many Group Rekeying

6.2.1. Protection of Rekeying Messages

When using a group rekeying scheme relying on one-to-many rekeying messages, the actual data content of each rekeying message is prepared according to what the rekeying scheme prescribes.

The following describes one possible method for the KDC to protect the rekeying messages when using the administrative keying material.

The method assumes that the following holds for the administrative keying material specified in the 'mgt_key_material' parameter of the Join Response (see Section 4.3.1).

* The encryption algorithm SHOULD be the same one used to protect communications in the group.

* The included symmetric encryption keys are accompanied by a corresponding and unique key identifier assigned by the KDC.

* A Base IV is also included, with the same size of the AEAD nonce considered by the encryption algorithm to use.

First, the KDC computes a COSE_Encrypt0 object as follows.

* The encryption key to use is selected from the administrative keying material, as defined by the rekeying scheme used in the group.

* The plaintext is the actual data content of the rekeying message.

* The Additional Authenticated Data (AAD) is empty, unless otherwise specified by separate documents profiling the use of the group rekeying scheme.

* Since the KDC is the only sender of rekeying messages, the AEAD nonce can be computed as follows, where NONCE_SIZE is the size in bytes of the AEAD nonce. Separate documents profiling the use of the group rekeying scheme may define alternative ways to compute the AEAD nonce.

  The KDC considers the following values.

  - COUNT, as a 2-byte unsigned integer associated with the used encryption key. Its value is set to 0 when starting to perform a new group rekeying instance, and is incremented after each use of the encryption key.

  - NEW_NUM, as the version number of the new group keying material to distribute in this rekeying instance, left-padded with zeros to exactly NONCE_SIZE - 2 bytes.

  Then, the KDC computes a Partial IV as the byte string concatenation of COUNT and NEW_NUM, in this order. Finally, the AEAD nonce is computed as the XOR between the Base IV and the Partial IV.

  In order to comply with the security requirements of AEAD encryption algorithms, the KDC MUST NOT reuse the same pair (AEAD encryption key, AEAD nonce). For example, this includes not using the same encryption key from the administrative keying material more than 2^16 times during the same rekeying instance.

* The protected header of the COSE_Encrypt0 object MUST include the following parameters.

  - 'alg', specifying the used encryption algorithm.

  - 'kid', specifying the identifier of the encryption key from the administrative keying material used to protect this rekeying message.

* The unprotected header of the COSE_Encrypt0 object MUST include the 'Partial IV' parameter, with value the Partial IV computed above.

In order to ensure source authentication, each rekeying message protected with the administrative keying material MUST be signed by the KDC. To this end, the KDC computes a countersignature of the COSE_Encrypt0 object, as described in Sections 3.2 and 3.3 of [RFC9338]. In particular, the following applies when computing the countersignature.

* The Countersign_structure contains the context text string "CounterSignature0".

* The private key of the KDC is used as signing key.

* The payload is the ciphertext of the COSE_Encrypt0 object.

* The Additional Authenticated Data (AAD) is empty, unless otherwise specified by separate documents profiling the use of a group rekeying scheme.

* The protected header of the signing object MUST include the parameter 'alg', specifying the used signature algorithm.

If source authentication of messages exchanged in the group is also ensured by means of signatures, then rekeying messages MUST be signed using the same signature algorithm and related parameters. Also, the KDC's authentication credential including the public key to use for signature verification MUST be provided in the Join Response through the 'kdc_cred' parameter, together with the corresponding proof-of-possession (PoP) evidence in the 'kdc_cred_verify' parameter.

If source authentication of messages exchanged in the group is not ensured by means of signatures, then the administrative keying material conveyed in the 'mgt_key_material' parameter of the Join Response sent by KDC (see Section 4.3.1) MUST also comprise a KDC's authentication credential including the public key to use for signature verification, together with a corresponding PoP evidence. Within the 'mgt_key_material' parameter, it is RECOMMENDED to specify this information by using the same format and encoding used for the parameters 'kdc_cred', 'kdc_nonce', and 'kdc_cred_verify' in the Join Response. It is up to separate documents profiling the use of the group rekeying scheme to specify such details.

After that, the KDC specifies the computed countersignature in the 'Countersignature0 version 2' header parameter of the COSE_Encrypt0 object.

Finally, the KDC specifies the COSE_Encrypt0 object as payload of a CoAP request, which is sent to the target group members as per the used message delivery method.

6.3. Misalignment of Group Keying Material

A group member can receive a message shortly after the group has been rekeyed, and new keying material has been distributed by the KDC. In the following two cases, this may result in misaligned keying material between the group members.

In the first case, the sender protects a message using the old group keying material. However, the recipient receives the message after having received the new group keying material, hence not being able to correctly process it. A possible way to limit the impact of this issue is to preserve the old, recent group keying material for a maximum amount of time defined by the application, during which it is used solely for processing incoming messages. By doing so, the recipient can still temporarily process received messages also by using the old, retained group keying material. Note that a former (compromised) group member can take advantage of this by sending messages protected with the old, retained group keying material. Therefore, a conservative application policy should not admit the storage of old group keying material. Eventually, the sender will have obtained the new group keying material too, and can possibly re-send the message protected with such keying material.

In the second case, the sender protects a message using the new group keying material, but the recipient receives that message before having received the new group keying material. Therefore, the recipient would not be able to correctly process the message and hence discards it. If the recipient receives the new group keying material shortly after that and the application at the sender endpoint performs retransmissions, the former will still be able to receive and correctly process the message. In any case, the recipient should actively ask the KDC for the latest group keying material according to an application-defined policy, for instance after a given number of unsuccessfully decrypted incoming messages.

7. Extended Scope Format

This section defines an extended format of binary encoded scope, which additionally specifies the semantics used to express the same access control information from the corresponding original scope.

As also discussed in Section 3.2, this enables a Resource Server to unambiguously process a received access token, also in case the Resource Server runs multiple applications or application profiles that involve different scope semantics.

The extended format is intended only for the 'scope' claim of access tokens, for the cases where the claim takes as value a CBOR byte string. That is, the extended format does not apply to the 'scope' parameter included in ACE messages, i.e., the Authorization Request and Authorization Response exchanged between the Client and the Authorization Server (see Sections 5.8.1 and 5.8.2 of [RFC9200]), the AS Request Creation Hints message from the Resource Server (see Section 5.3 of [RFC9200]), and the Introspection Response from the Authorization Server (see Section 5.9.2 of [RFC9200]).

The value of the 'scope' claim following the extended format is composed as follows. Given the original scope using a semantics SEM and encoded as a CBOR byte string, the corresponding extended scope consists of the same CBOR byte string enclosed by a CBOR tag [RFC8949], whose tag number identifies the semantics SEM.

The resulting tagged CBOR byte string is used as value of the 'scope' claim of the access token.

Figure 40 and Figure 41 build on the examples in Section 3.2, and show the corresponding extended scopes.

```
;# include rfc9237

gname = tstr

permissions = uint .bits roles

roles = &(
   Requester: 1,
   Responder: 2,
   Monitor: 3,
   Verifier: 4
)

scope_entries = AIF-Generic<gname, permissions>

scope = bstr .cbor scope_entries

extended_scope = #6.TAG_FOR_THIS_SEMANTICS(scope)
```

Figure 40: Example CDDL definition of scope, using the default Authorization Information Format

```
gname = tstr

role = tstr

scope_entry = [ gname , ? ( role / [ 2*role ] ) ]

scope_entries = [ * scope_entry ]

scope = bstr .cbor scope_entries

extended_scope = #6.TAG_FOR_THIS_SEMANTICS(scope)
```

Figure 41: CDDL definition of scope, using as example group name encoded as tstr and role as tstr

The usage of the extended scope format is not limited to application profiles of this specification or to applications based on group communication. Rather, it is generally applicable to any application and application profile where access control information in the access token is expressed as a binary encoded scope.

Applications and application profiles using the extended format of scope have to specify which CBOR tag from [CBOR.Tags] is used for identifying the scope semantics, or to register a new CBOR tag if a suitable one does not exist already (REQ28). In case there is an already existing, suitable CBOR tag, a new CBOR tag should not be registered in order to avoid codepoint squatting.

If the binary encoded scope uses a semantics associated with a registered CoAP Content-Format [RFC7252][CoAP.Content.Formats], then a suitable CBOR tag associated with that CoAP Content-Format would already be registered, as defined in Section 4.3 of [RFC9277].

This is especially relevant when the binary encoded scope uses the AIF format. That is, it is expected that the definition of an AIF specific data model comes together with the registration of CoAP Content-Formats for the relevant combinations of its Toid and Tperm values. As discussed above, this yields the automatic registration of the CBOR tags associated with those CoAP Content-Formats.

8. ACE Groupcomm Parameters

This specification defines a number of parameters used during the second part of the message exchange, after the exchange of Token Transfer Request and Response. The table below summarizes them, and specifies the CBOR key to use instead of the full descriptive name.

Note that the media type application/ace-groupcomm+cbor MUST be used when these parameters are transported in the respective message fields.

Name	CBOR Key	CBOR Type	Reference
gid	0	array	[RFC-XXXX]
gname	1	array of tstr	[RFC-XXXX]
guri	2	array of tstr	[RFC-XXXX]
scope	3	bstr	[RFC-XXXX]
get_creds	4	array / Simple value "null"	[RFC-XXXX]
client_cred	5	bstr	[RFC-XXXX]
cnonce	6	bstr	[RFC-XXXX]
client_cred_verify	24	bstr	[RFC-XXXX]
creds_repo	25	tstr	[RFC-XXXX]
control_uri	26	tstr	[RFC-XXXX]
gkty	7	int / tstr	[RFC-XXXX]
key	8	See the "ACE Groupcomm Key Types" registry	[RFC-XXXX]
num	9	int	[RFC-XXXX]
ace_groupcomm_profile	10	int	[RFC-XXXX]
exp	11	uint	[RFC-XXXX]
exi	12	uint	[RFC-XXXX]
creds	13	array	[RFC-XXXX]
peer_roles	14	array	[RFC-XXXX]
peer_identifiers	15	array	[RFC-XXXX]

group_policies	16	map	[RFC-XXXX]
kdc_cred	17	bstr	[RFC-XXXX]
kdc_nonce	18	bstr	[RFC-XXXX]
kdc_cred_verify	19	bstr	[RFC-XXXX]
rekeying_scheme	20	int	[RFC-XXXX]
mgt_key_material	27	bstr	[RFC-XXXX]
control_group_uri	28	tstr	[RFC-XXXX]
sign_info	29	array / Simple value "null"	[RFC-XXXX]
kdcchallenge	30	bstr	[RFC-XXXX]

Figure 42: ACE Groupcomm Parameters

Note to RFC Editor: In Figure 42, please replace all occurrences of "[RFC-XXXX]" with the RFC number of this specification and delete this paragraph.

The KDC is expected to support all the parameters above. Instead, a Client can support only a subset of such parameters, depending on the roles it expects to take in the joined groups or on other conditions defined in application profiles of this specification.

In the following, the parameters are categorized according to the support expected by Clients. That is, a Client that supports a parameter is able to: i) use and specify it in a request message to the KDC; and ii) understand and process it if specified in a response message from the KDC. It is REQUIRED of application profiles of this specification to sort their newly defined parameters according to the same categorization (REQ29).

Note that the actual use of a parameter and its inclusion in a message depends on the specific exchange, the specific Client and group involved, as well as what is defined in the used application profile of this specification.

A Client MUST support the following parameters.

* 'scope', 'cnonce', 'gkty', 'key', 'num', 'exp', 'exi', 'gid', 'gname', 'guri', 'creds', 'peer_identifiers', 'ace_groupcomm_profile', 'control_uri', 'rekeying_scheme'.

A Client SHOULD support the following parameter.

* 'get_creds'. That is, not supporting this parameter would yield the inconvenient and undesirable behavior where: i) the Client does not ask for the other group members' authentication credentials upon joining the group (see Section 4.3.1.1); and ii) later on as a group member, the Client only retrieves the authentication credentials of all group members (see Section 4.4.2.1).

The following conditional parameters are relevant only if specific conditions hold. It is REQUIRED of application profiles of this specification to define whether Clients must, should, or may support these parameters, and under which circumstances (REQ30).

* 'client_cred' and 'client_cred_verify'. These parameters are relevant for a Client that has an authentication credential to use in a joined group.

* 'kdcchallenge'. This parameter is relevant for a Client that has an authentication credential to use in a joined group and that provides the access token to the KDC through a Token Transfer Request (see Section 3.3).

* 'creds_repo'. This parameter is relevant for a Client that has an authentication credential to use in a joined group and that makes it available from a key repository different than the KDC.

* 'group_policies'. This parameter is relevant for a Client that is interested in the specific policies used in a group, but it does not know them or cannot become aware of them before joining that group.

* 'peer_roles'. This parameter is relevant for a Client that has to know about the roles of other group members, especially when retrieving and handling their corresponding authentication credentials.

* 'kdc_nonce', 'kdc_cred', 'kdc_cred_verify'. These parameters are relevant for a Client that joins a group for which, as per the used application profile of this specification, the KDC has an associated authentication credential and this is required for the correct group operation.

* 'mgt_key_material'. This parameter is relevant for a Client that supports an advanced rekeying scheme possibly used in the group, such as based on one-to-many rekeying messages sent over IP multicast.

* 'control_group_uri'. This parameter is relevant for a Client that supports the hosting of local resources each associated with a group (hence acting as CoAP server) and the reception of one-to-many requests sent to those resources by the KDC (e.g., over IP multicast), targeting multiple members of the corresponding group. Examples of related management operations that the KDC can perform by this means are the eviction of group members and the execution of a group rekeying process through an advanced rekeying scheme, such as based on one-to-many rekeying messages.

9. ACE Groupcomm Error Identifiers

This specification defines a number of values that the KDC can use as error identifiers. These are used in error responses with Content-Format application/concise-problem-details+cbor, as values of the 'error-id' field within the Custom Problem Detail entry 'ace-groupcomm-error' (see Section 4.1.2).

Value	Description
0	Operation permitted only to group members
1	Request inconsistent with the current roles
2	Authentication credential incompatible with the group configuration
3	Invalid proof-of-possession evidence
4	No available node identifiers
5	Group membership terminated
6	Group deleted

Figure 43: ACE Groupcomm Error Identifiers

If a Client supports the problem-details format [RFC9290] and the Custom Problem Detail entry 'ace-groupcomm-error' defined in Section 4.1.2, and is able to understand the error specified in the 'error-id' field therein, then the Client can use that information to

determine what actions to take next. If the Concise Problem Details data item specified in the error response includes the 'detail' entry and the Client supports it, such an entry may provide additional context.

In particular, the following guidelines apply, and application profiles of this specification can define more detailed actions for the Client to take when learning that a specific error has occurred.

* In case of error 0, the Client should stop sending the request in question to the KDC. Rather, the Client should first join the targeted group. If it has not happened already, this first requires the Client to obtain an appropriate access token authorizing access to the group and provide it to the KDC.

* In case of error 1, the Client as a group member should re-join the group with all the roles needed to perform the operation in question. This might require the Client to first obtain a new access token and provide it to the KDC, if the current access token does not authorize it to take those roles in the group. For operations admitted to a Client which is not a group member (e.g., an external signature verifier), the Client should first obtain a new access token authorizing to also have the missing roles.

* In case of error 2, the Client has to obtain or self-generate a different asymmetric key pair, as aligned to the public key algorithms and parameters used in the targeted group. After that, the Client should provide the KDC with its new authentication credential, consistent with the format used in the targeted group and including the new public key.

* In case of error 3, the Client should ensure to be computing its proof-of-possession evidence by correctly using the parameters and procedures defined in the used application profile of this specification. In an unattended setup, it might be not possible for a Client to autonomously diagnose the error and take an effective next action to address it.

* In case of error 4, the Client should wait for a certain (pre-configured) amount of time, before trying re-sending its request to the KDC.

* In case of error 5, the Client may try joining the group again. This might require the Client to first obtain a new access token and provide it to the KDC, e.g., if the current access token has expired.

* In case of error 6, the Client should clean up its state regarding the group, just like if it has left the group with no intention to re-join it.

10. Security Considerations

Security considerations are inherited from the ACE framework [RFC9200], and from the specific transport profile of ACE used between the Clients and the KDC, e.g., [RFC9202] and [RFC9203].

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.

Furthermore, the following security considerations apply.

10.1. Secure Communication in the Group

When a group member receives a message from a certain sender for the first time since joining the group, it needs to have a mechanism in place to avoid replayed messages and to assert their freshness, e.g., Appendix B.1.2 of [RFC8613] or Section 10 of [I-D.ietf-core-oscore-groupcomm]. Such a mechanism aids the recipient group member also in case it has rebooted and lost the security state used to protect previous group communications with that sender.

By its nature, the KDC is invested with a large amount of trust, since it acts as a generator and provider of the symmetric keying material used to protect communications in each of its groups. While details depend on the specific communication and security protocols used in the group, the KDC is in the position to decrypt messages exchanged in the group as if it was also a group member, as long as those are protected through commonly shared group keying material.

A compromised KDC would thus put the attacker in the same position, which also means that:

* The attacker can generate and control new group keying material, hence possibly rekeying the group and evicting certain group members as part of a broader attack.

* The attacker can actively participate in communications in a group even without been authorized to join it, and can allow further unauthorized entities to do so.

* The attacker can build erroneous associations between node identifiers and group members' authentication credentials.

On the other hand, as long as the security protocol used in the group ensures source authentication of messages (e.g., by means of signatures), the KDC is not able to impersonate group members since it does not have their private keys.

Further security considerations are specific to the communication and security protocols used in the group, and thus have to be provided by those protocols and complemented by the application profiles of this specification using them.

10.2. Update of Group Keying Material

The KDC can generate new group keying material and provide it to the group members (rekeying) through the rekeying scheme used in the group, as discussed in Section 6.

In particular, the KDC must renew the group keying material latest upon its expiration. Before then, the KDC MAY also renew the group keying material on a regular or periodical fashion.

Unless otherwise defined by an application profile of this specification, the KDC SHOULD renew the group keying material upon a group membership change. As a possible exception, the KDC may not rekey the group upon the joining of a new group member, if the application does not require backward security. As another possible exception discussed more in detail later in this section, the KDC may rely on a rekeying policy that reasonably take into account the expected rate of group membership changes and the duration of a group rekeying.

Since the minimum number of group members is one, the KDC SHOULD provide even a Client joining an empty group with new keying material never used before in that group. Similarly, the KDC SHOULD provide new group keying material also to a Client that remains the only member in the group after the leaving of other group members.

Note that the considerations in Section 10.1 about dealing with replayed messages still hold, even in case the KDC rekeys the group upon every single joining of a new group member. However, if the KDC has renewed the group keying material upon a group member’s joining, and the time interval between the end of the rekeying process and that member’s joining is sufficiently small, then that group member is also on the safe side, since it would not accept replayed messages protected with the old group keying material previous to its joining.

Once a joining node has obtained the new, latest keying material through a Join Response from the KDC (see Section 4.3.1.1), the joining node becomes able to read any message that was exchanged in

the group and protected with that keying material. This is the case if the KDC provides the current group members with the new, latest keying material before completing the joining procedure. However, the joining node is not able to read messages exchanged in the group and protected with keying material older than the one provided in the Join Response, i.e., having a strictly lower version number NUM.

A node that has left the group should not expect any of its outgoing messages to be successfully processed, if received by other nodes in the group after its leaving, due to a possible group rekeying occurred before the message reception.

The KDC may enforce a rekeying policy that takes into account the overall time required to rekey the group, as well as the expected rate of changes in the group membership. That is, the KDC may not rekey the group at each and every group membership change, for instance if members' joining and leaving occur frequently and performing a group rekeying takes too long. Instead, the KDC might rekey the group after a minimum number of group members have joined or left within a given time interval, or after a maximum amount of time since the last group rekeying was completed, or yet during predictable network inactivity periods.

However, this would result in the KDC not constantly preserving backward and forward security in the group. That is:

* Newly joining group members would be able to access the keying material used before their joining, and thus they could access past group communications if they have recorded old exchanged messages. This might still be acceptable for some applications and in situations where the new group members are freshly deployed through strictly controlled procedures.

* The leaving group members would remain able to access upcoming group communications, as protected with the current keying material that has not been updated. This is typically undesirable, especially if the leaving group member is compromised or suspected to be, and it might have an impact or compromise the security properties of the protocols used in the group to protect messages exchanged among the group members.

The KDC should renew the group keying material in case it has rebooted, even if it stores the whole group keying material in persistent storage. This assumes that the secure associations with the current group members as well as any administrative keying material required to rekey the group are also stored in persistent storage.

However, if the KDC relies on Observe notifications to distribute the new group keying material, the KDC would have lost all the current ongoing Observations with the group members after rebooting, and the group members would continue using the old group keying material. Therefore, the KDC will rather rely on each group member asking for the new group keying material (see Section 4.3.2.1 and Section 4.8.1.1), or rather perform a group rekeying by actively sending rekeying messages to group members as discussed in Section 6.

The KDC needs to have a mechanism in place to detect DoS attacks from nodes repeatedly performing actions that might trigger a group rekeying. Such actions can include leaving and/or re-joining the group at high rates, or often asking the KDC for new individual keying material. Ultimately, the KDC can resort to removing these nodes from the group and (temporarily) preventing them from joining the group again.

The KDC also needs to have a congestion control mechanism in place, in order to avoid network congestion upon distributing new group keying material. For example, CoAP and Observe give guidance on such mechanisms, see Section 4.7 of [RFC7252] and Section 4.5.1 of [RFC7641].

10.3. Block-Wise Considerations

If the Block-Wise CoAP options [RFC7959] are used, and the keying material is updated in the middle of a Block-Wise transfer, the sender of the blocks just changes the group keying material to the updated one and continues the transfer. As long as both sides get the new group keying material, updating the group keying material in the middle of a transfer will not cause any issue. Otherwise, the sender will have to transmit the message again, when receiving an error message from the recipient.

Compared to a scenario where the transfer does not use Block-Wise, depending on how fast the group keying material is changed, the group members might consume a larger amount of the network bandwidth by repeatedly resending the same blocks, which might be problematic.

11. 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.

11.1. Media Type Registrations

This specification registers the 'application/ace-groupcomm+cbor' media type for messages of the protocols defined in this document following the ACE exchange and carrying parameters encoded in CBOR. This registration follows the procedures specified in [RFC6838].

Type name: application

Subtype name: ace-groupcomm+cbor

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Must be encoded as a CBOR map containing the parameters defined in [RFC-XXXX].

Security considerations: See Section 10 of [RFC-XXXX].

Interoperability considerations: N/A

Published specification: [RFC-XXXX]

Applications that use this media type: The type is used by Authorization Servers, Clients, and Resource Servers that support the ACE groupcomm framework as specified in [RFC-XXXX].

Fragment identifier considerations: N/A

Additional information: N/A

Person & email address to contact for further information: ACE WG mailing list (ace@ietf.org) or IETF Applications and Real-Time Area (art@ietf.org)

Intended usage: COMMON

Restrictions on usage: None

Author/Change controller: IETF

Provisional registration: No

11.2. CoAP Content-Formats

IANA is asked to register the following entry to the "CoAP Content-Formats" registry within the "CoRE Parameters" registry group.

Content Type: application/ace-groupcomm+cbor

Content Coding: -

ID: TBD

Reference: [RFC-XXXX]

11.3. OAuth Parameters

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

* Parameter name: sign_info

* Parameter usage location: client-rs request, rs-client response

* Change Controller: IETF

* Specification Document(s): [RFC-XXXX]

* Parameter name: kdcchallenge

* Parameter usage location: rs-client response

* Change Controller: IETF

* Specification Document(s): [RFC-XXXX]

11.4. OAuth Parameters CBOR Mappings

IANA is asked to register the following entries in the "OAuth Parameters CBOR Mappings" registry following the procedure specified in Section 8.10 of [RFC9200].

* Name: sign_info

* CBOR Key: TBD (range -256 to 255)

* Value Type: Simple value "null" / array

* Reference: [RFC-XXXX]

* Name: kdcchallenge

* CBOR Key: TBD (range -256 to 255)

* Value Type: Byte string

* Reference: [RFC-XXXX]

11.5. Interface Description (if=) Link Target Attribute Values

IANA is asked to register the following entry in the "Interface Description (if=) Link Target Attribute Values" registry within the "CoRE Parameters" registry group.

* Value: ace.groups

* Description: The KDC interface at the parent resource of group-membership resources is used to retrieve names of security groups using the ACE framework.

* Reference: Section 4.1 of [RFC-XXXX]

* Value: ace.group

* Description: The KDC interface at a group-membership resource is used to provision keying material and related information and policies to members of the corresponding security group using the ACE framework.

* Reference: Section 4.1 of [RFC-XXXX]

11.6. Custom Problem Detail Keys Registry

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

* Key Value: TBD

* Name: ace-groupcomm-error

* Brief Description: Carry [RFC-XXXX] problem details in a Concise Problem Details data item.

* Change Controller: IETF

* Reference: Section 4.1.2 of [RFC-XXXX]

11.7. ACE Groupcomm Parameters

This specification establishes the "ACE Groupcomm Parameters" IANA registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group.

The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.14. Values in this registry are covered by different registration policies as indicated. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

* Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

* CBOR Key: This is the value used as the CBOR key of the item. These values MUST be unique. The value can be a positive integer, a negative integer, or a text string. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 as well as text strings of length 1 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535, as well as text strings of length 2 are designated as "Specification Required". Integer values greater than 65535 as well as text strings of length greater than 2 are designated as "Expert Review". Integer values less than -65536 are marked as "Private Use".

* CBOR Type: This contains the CBOR type of the item, or a pointer to the registry that defines its type, when that depends on another item.

* Reference: This contains a pointer to the public specification for the item.

This registry has been initially populated with the values in Figure 42.

11.8. ACE Groupcomm Key Types

This specification establishes the "ACE Groupcomm Key Types" IANA registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group.

The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.14. Values in this registry are covered by different registration policies as indicated. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

* Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

* Key Type Value: This is the value used to identify the keying material. These values MUST be unique. The value can be a positive integer, a negative integer, or a text string. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 as well as text strings of length 1 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535, as well as text strings of length 2 are designated as "Specification Required". Integer values greater than 65535 as well as text strings of length greater than 2 are designated as "Expert Review". Integer values less than -65536 are marked as "Private Use".

* Profile: This field may contain one or more descriptive strings of application profiles to be used with this item. The values should be taken from the Name column of the "ACE Groupcomm Profiles" registry.

* Description: This field contains a brief description of the keying material.

* Reference: This contains a pointer to the public specification for the format of the keying material, if one exists.

This registry has been initially populated with the value in Figure 11.

11.9. ACE Groupcomm Profiles

This specification establishes the "ACE Groupcomm Profiles" IANA registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group.

The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.14. Values in this registry are covered by different registration policies as indicated. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

* Name: The name of the application profile, to be used as value of the profile attribute.

* Description: Text giving an overview of the application profile and the context it is developed for.

* CBOR Value: CBOR abbreviation for the name of this application profile. These values MUST be unique. The value can be a positive integer or a negative integer. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535 are designated as "Specification Required". Integer values greater than 65535 are designated as "Expert Review". Integer values less than -65536 are marked as "Private Use".

* Reference: This contains a pointer to the public specification of the abbreviation for this application profile, if one exists.

This registry has been initially populated with the value in Figure 12.

11.10. ACE Groupcomm Policies

This specification establishes the "ACE Groupcomm Policies" IANA registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group.

The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.14. Values in this registry are covered by different registration policies as indicated. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

* Name: The name of the group communication policy.

* CBOR label: The value to be used to identify this group communication policy. These values MUST be unique. The value can be a positive integer, a negative integer, or a text string. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 as well as text strings of length 1 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535, as well as text strings of length 2 are designated as "Specification Required". Integer values greater than 65535 as well as text strings of length greater than 2 are designated as "Expert Review". Integer values less than -65536 are marked as "Private Use".

* CBOR type: the CBOR type used to encode the value of this group communication policy.

* Description: This field contains a brief description for this group communication policy.

* Reference: This field contains a pointer to the public specification providing the format of the group communication policy, if one exists.

This registry has been initially populated with the values in Figure 13.

11.11. Sequence Number Synchronization Methods

This specification establishes the "Sequence Number Synchronization Methods" IANA registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group.

The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.14. Values in this registry are covered by different registration policies as indicated. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

* Name: The name of the sequence number synchronization method.

* Value: The value to be used to identify this sequence number synchronization method. These values MUST be unique. The value can be a positive integer, a negative integer, or a text string. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 as well as text strings of length 1 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535, as well as text strings of length 2 are designated as "Specification Required". Integer values greater than 65535 as well as text strings of length greater than 2 are designated as "Expert Review". Integer values less than -65536 are marked as "Private Use".

* Description: This field contains a brief description for this sequence number synchronization method.

* Reference: This field contains a pointer to the public specification describing the sequence number synchronization method.

11.12. ACE Groupcomm Errors

This specification establishes the "ACE Groupcomm Errors" IANA registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group.

The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.14. Values in this registry are covered by different registration policies as indicated. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

* Value: The value to be used to identify the error. These values MUST be unique. The value can be a positive integer or a negative integer. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535 are designated as "Specification Required". Integer values greater than 65535 are designated as "Expert Review". Integer values less than -65536 are marked as "Private Use".

* Description: This field contains a brief description of the error.

* Reference: This field contains a pointer to the public specification defining the error, if one exists.

This registry has been initially populated with the values in Figure 43. The Reference column for all of these entries refers to this document.

11.13. ACE Groupcomm Rekeying Schemes

This specification establishes the "ACE Groupcomm Rekeying Schemes" IANA registry within the "Authentication and Authorization for Constrained Environments (ACE)" registry group.

The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 11.14. Values in this registry are covered by different registration policies as indicated. It should be noted that, in addition to the expert review, some portions of the registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this registry are:

* Value: The value to be used to identify the group rekeying scheme. These values MUST be unique. The value can be a positive integer or a negative integer. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as "Standards Action With Expert Review". Integer values from -65536 to -257 and from 256 to 65535 are designated as "Specification Required". Integer values greater than 65535 are designated as "Expert Review". Integer values less than -65536 are marked as "Private Use".

* Name: The name of the group rekeying scheme.

* Description: This field contains a brief description of the group rekeying scheme.

* Reference: This field contains a pointer to the public specification defining the group rekeying scheme, if one exists.

This registry has been initially populated with the value in Figure 14.

11.14. Expert Review Instructions

The IANA Registries established in this document are defined as expert review. This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

* Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. The zones tagged as private use are intended for testing purposes and closed environments, code points in other ranges should not be assigned for testing.

* Specifications are required for the standards track range of point assignment. Specifications should exist for specification required ranges, but early assignment before a specification is available is considered to be permissible. When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for.

* Experts should take into account the expected usage of fields when approving point assignments. The fact that there is a range for Standards Track documents does not mean that a Standards Track document cannot have points assigned outside of that range. The length of the encoded value should be weighed against how many code points of that length are left, the size of device it will be used on, and the number of code points left that encode to that size.

12. References

12.1. Normative References

[CBOR.Tags]
IANA, "Concise Binary Object Representation (CBOR) Tags", <https://www.iana.org/assignments/cbor-tags/cbor-tags.xhtml>.

[CoAP.Content.Formats]
IANA, "CoAP Content-Formats", <https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#content-formats>.

[COSE.Algorithms]
IANA, "COSE Algorithms",
<https://www.iana.org/assignments/cose/
cose.xhtml#algorithms>.

[COSE.Header.Parameters]
IANA, "COSE Header Parameters",
<https://www.iana.org/assignments/cose/cose.xhtml#header-
parameters>.

[COSE.Key.Types]
IANA, "COSE key Types",
<https://www.iana.org/assignments/cose/cose.xhtml#key-
type>.

[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>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link
Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,
<https://www.rfc-editor.org/rfc/rfc6690>.

[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>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type
Specifications and Registration Procedures", BCP 13,
RFC 6838, DOI 10.17487/RFC6838, January 2013,
<https://www.rfc-editor.org/rfc/rfc6838>.

[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>.

[RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T.
Bose, "Constrained Application Protocol (CoAP) Option for
No Server Response", RFC 7967, DOI 10.17487/RFC7967,
August 2016, <https://www.rfc-editor.org/rfc/rfc7967>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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>.

[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>.

[RFC9237] Bormann, C., "An Authorization Information Format (AIF) for Authentication and Authorization for Constrained Environments (ACE)", RFC 9237, DOI 10.17487/RFC9237, August 2022, <https://www.rfc-editor.org/rfc/rfc9237>.

[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>.

[RFC9338] Schaad, J., "CBOR Object Signing and Encryption (COSE): Countersignatures", STD 96, RFC 9338, DOI 10.17487/RFC9338, December 2022, <https://www.rfc-editor.org/rfc/rfc9338>.

12.2. Informative References

[I-D.ietf-core-coap-pubsub]
Jimenez, J., Koster, M., and A. Keränen, "A publish-subscribe architecture for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-coap-pubsub-13, 20 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-coap-pubsub-13>.

[I-D.ietf-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-10, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-10>.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and J. Park, "Group Object Security for Constrained RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-20, 2 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-20>.

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-07, 20 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-07>.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsüss, C., and P. Van der Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-

discovery-14, 8 September 2023,
<https://datatracker.ietf.org/doc/html/draft-tiloca-core-oscore-discovery-14>.

[RFC2093] Harney, H. and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification", RFC 2093, DOI 10.17487/RFC2093, July 1997, <https://www.rfc-editor.org/rfc/rfc2093>.

[RFC2094] Harney, H. and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture", RFC 2094, DOI 10.17487/RFC2094, July 1997, <https://www.rfc-editor.org/rfc/rfc2094>.

[RFC2627] Wallner, D., Harder, E., and R. Agee, "Key Management for Multicast: Issues and Architectures", RFC 2627, DOI 10.17487/RFC2627, June 1999, <https://www.rfc-editor.org/rfc/rfc2627>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[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>.

[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>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/rfc/rfc7641>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/rfc/rfc7959>.

[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>.

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

[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>.

[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>.

Appendix A. Requirements for Application Profiles

This section lists the requirements for application profiles of this specification, for the convenience of application profile designers.

A.1. Mandatory-to-Address Requirements

* REQ1: Specify the format and encoding of 'scope'. This includes defining the set of possible roles and their identifiers, as well as the corresponding encoding to use in the scope entries according to the used scope format (see Section 3.1).

* REQ2: If the AIF format of 'scope' is used, register its specific instance of "Toid" and "Tperm" as Media Type parameters and a corresponding Content-Format, as per the guidelines in [RFC9237].

* REQ3: If used, specify the acceptable values for 'sign_alg' (see Section 3.3).

* REQ4: If used, specify the acceptable values for 'sign_parameters' (see Section 3.3).

* REQ5: If used, specify the acceptable values for 'sign_key_parameters' (see Section 3.3).

* REQ6: Specify the acceptable formats for authentication credentials and, if used, the acceptable values for 'cred_fmt' (see Section 3.3).

* REQ7: If the value of the GROUPNAME URI path and the group name in the access token scope (gname in Section 3.2) are not required to coincide, specify the mechanism to map the GROUPNAME value in the URI to the group name (see Section 4.1).

* REQ8: Define whether the KDC has an authentication credential and if this has to be provided through the 'kdc_cred' parameter, see Section 4.3.1.

* REQ9: Specify if any part of the KDC interface as defined in this document is not supported by the KDC (see Section 4.1).

* REQ10: Register a Resource Type for the group-membership resource, which is used to discover the correct URL for sending a Join Request to the KDC (see Section 4.1).

* REQ11: Define what specific actions (e.g., CoAP methods) are allowed on each resource provided by the KDC interface, depending on whether the Client is a current group member; the roles that a Client is authorized to take as per the obtained access token (see Section 3.1); and the roles that the Client has as current group member.

* REQ12: Categorize possible newly defined operations for Clients into primary operations expected to be minimally supported and secondary operations, and provide accompanying considerations (see Section 4.1.1).

* REQ13: Specify the encoding of group identifier (see Section 4.2.1).

* REQ14: Specify the approaches used to compute and verify the PoP evidence to include in 'client_cred_verify', and which of those approaches is used in which case (see Section 4.3.1).

* REQ15: Specify how the nonce N_S is generated, if the token is not provided to the KDC through the Token Transfer Request to the authz-info endpoint (e.g., if it is used directly to validate TLS instead).

* REQ16 Define the initial value of the 'num' parameter (see Section 4.3.1).

* REQ17: Specify the format of the 'key' parameter and register a corresponding entry in the "ACE Groupcomm Key Types" IANA registry (see Section 4.3.1).

* REQ18: Specify the acceptable values of the 'gkty' parameter (see Section 4.3.1).

* REQ19: Specify and register the application profile identifier (see Section 4.3.1).

* REQ20: If used, specify the format and content of 'group_policies' and its entries. Specify the policies default values (see Section 4.3.1).

* REQ21: Specify the approaches used to compute and verify the PoP evidence to include in 'kdc_cred_verify', and which of those approaches is used in which case (see Section 4.3.1 and Section 4.5.1). If external signature verifiers are supported, specify how those provide a nonce to the KDC to be used for computing the PoP evidence (see Section 4.5.1).

* REQ22: Specify the communication protocol that the members of the group must use (e.g., CoAP for group communication).

* REQ23: Specify the security protocol the group members must use to protect their communication (e.g., group OSCORE). This must provide encryption, integrity, and replay protection.

* REQ24: Specify how the communication is secured between Client and KDC. Optionally, specify a transport profile of ACE [RFC9200] to use between Client and KDC (see Section 4.3.1.1).

* REQ25: Specify the format of the identifiers of group members (see Section 4.3.1 and Section 4.4.1).

* REQ26: Specify policies at the KDC to handle ids that are not included in 'get_creds' (see Section 4.4.1).

* REQ27: Specify the format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label (see Section 4.8.1).

* REQ28: Specify which CBOR tag is used for identifying the semantics of binary scopes, or register a new CBOR tag if a suitable one does not exist already (see Section 7).

* REQ29: Categorize newly defined parameters according to the same criteria of Section 8.

* REQ30: Define whether Clients must, should, or may support the conditional parameters defined in Section 8, and under which circumstances.

A.2. Optional-to-Address Requirements

* OPT1: Optionally, if the textual format of 'scope' is used, specify CBOR values to use for abbreviating the role identifiers in the group (see Section 3.1).

* OPT2: Optionally, specify the additional parameters used in the exchange of Token Transfer Request and Response (see Section 3.3).

* OPT3: Optionally, specify the negotiation of parameter values for signature algorithm and signature keys, if 'sign_info' is not used (see Section 3.3).

* OPT4: Optionally, specify possible or required payload formats for specific error cases.

* OPT5: Optionally, specify additional identifiers of error types, as values of the 'error-id' field within the Custom Problem Detail entry 'ace-groupcomm-error' (see Section 4.1.2).

* OPT6: Optionally, specify the encoding of 'creds_repo' if the default is not used (see Section 4.3.1).

* OPT7: Optionally, specify the functionalities implemented at the 'control_uri' resource hosted at the Client, including message exchange encoding and other details (see Section 4.3.1).

* OPT8: Optionally, specify the behavior of the handler in case of failure to retrieve an authentication credential for the specific node (see Section 4.3.1).

* OPT9: Optionally, define a default group rekeying scheme, to refer to in case the 'rekeying_scheme' parameter is not included in the Join Response (see Section 4.3.1).

* OPT10: Optionally, specify the functionalities implemented at the 'control_group_uri' resource hosted at the Client, including message exchange encoding and other details (see Section 4.3.1).

* OPT11: Optionally, specify policies that instruct Clients to retain messages and for how long, if they are unsuccessfully decrypted (see Section 4.8.1.1). This makes it possible to decrypt such messages after getting updated keying material.

* OPT12: Optionally, specify for the KDC to perform group rekeying (together or instead of renewing individual keying material) when receiving a Key Renewal Request (see Section 4.8.2.1).

* OPT13: Optionally, specify how the identifier of a group member's authentication credential is included in requests sent to other group members (see Section 4.9.1.1).

* OPT14: Optionally, specify additional information to include in rekeying messages for the "Point-to-Point" group rekeying scheme (see Section 6).

Appendix B. Extensibility for Future COSE Algorithms

As defined in Section 8.1 of [RFC9053], future algorithms can be registered in the "COSE Algorithms" registry [COSE.Algorithms] as specifying none or multiple COSE capabilities.

To enable the seamless use of such future registered algorithms, this section defines a general, agile format for each 'sign_info_entry' of the 'sign_info' parameter in the Token Transfer Response, see Section 3.3.1.

If any of the currently registered COSE algorithms is considered, using this general format yields the same structure of 'sign_info_entry' defined in this document, thus ensuring backward compatibility.

B.1. Format of 'sign_info_entry'

The format of each 'sign_info_entry' (see Section 3.3.1) is generalized as follows.

* 'sign_parameters' includes N >= 0 elements, each of which is a COSE capability of the signature algorithm indicated in 'sign_alg'.

  In particular, 'sign_parameters' has the same format and value of the COSE capabilities array for the signature algorithm indicated in 'sign_alg', as specified for that algorithm in the 'Capabilities' column of the "COSE Algorithms" registry [COSE.Algorithms].

* 'sign_key_parameters' is replaced by N elements 'sign_capab', each of which is a CBOR array.

  The i-th 'sign_capab' array (i = 0, ..., N-1) is the array of COSE capabilities for the algorithm capability specified in 'sign_parameters'[i].

  In particular, each 'sign_capab' array has the same format and value of the COSE capabilities array for the algorithm capability specified in 'sign_parameters'[i].

  Such a COSE capabilities array is currently defined for the algorithm capability COSE key type, in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types].

```
sign_info_entry =
[
    id : gname / [ + gname ],
    sign_alg : int / tstr,
    sign_parameters : [ * alg_capab : any ],
  * sign_capab : [ * capab : any ],
    cred_fmt : int / null
]

gname = tstr
```

Figure 44: 'sign_info_entry' with general format

Appendix C. Document Updates

RFC EDITOR: PLEASE REMOVE THIS SECTION.

C.1. Version -17 to -18

* Provided more details when early introducing "backward security" and "forward security".

* Clarified definition and semantics of "group name" and "node name".

* Clarified definition of "individual keying material".

* Clarified definition of "Dispatcher".

* Enforced consistent use of leading slash in URI paths.

* Fixed CDDL definitions and examples in CBOR diagnostic notation.

* RFC 9290 is used instead of the custom format for error responses.

* Clarified which operations are limited to group members and which are allowed also to non group members.

* Improved examples of message exchange.

* Added ASCII-art diagrams with examples of group rekeying.

* Clarified for how long nonces are stored at the KDC.

* Clarified that the KDC might not have to store the 'cnonce' from a Join Request.

* Consistency fix: Clients always support the 'cnonce' parameter.

* Added new parameter 'exi' providing the residual lifetime of the current group keying material.

* Clarified text about the KDC knowledge of compromised nodes.

* Clarified the impact on performance of a one-to-many group rekeying.

* Mentioned explicit exceptions to a group rekeying at each group membership change.

* Explained reasons for delaying a rekeying and halting communications.

* Fixes in current IANA registrations.

* Added integer abbreviation values for registrations in new IANA registries.

* IANA registration of two CoRE if= values: "ace.group" and "ace.groups".

* Editorial fixes and improvements.

C.2. Version -16 to -17

* Expanded definition of "Dispatcher".

* Added definition of "Individual keying material" to the terminology.

* Early definition of "backward security" and "forward security".

* Clarified high-level breakdown of the key provisioning process in two phases.

* Fixed the CDDL definition of 'sign_info_entry'.

* Clarified meaning of the 'cred_fmt' and 'exp' parameters.

* Clarified that invariance applies to resources paths, not to resources.

* Relaxed rule about including the 'peer_roles' parameter.

* Ensured that the KDC always has a Client-side challenge for computing a proof-of-possession evidence.

* More guidelines for group members that fail to decrypt messages.

* Fetching the latest keying material can happen before the old, stored one expires.

* Renewing the current keying material can happen before it expires.

* Moved up the discussion on misalignment of group keying material.

* Expanded security considerations on group rekeying for joining nodes.

* Revised size of integer for building AEAD nonces for group rekeying.

* Added reserved value to the "ACE Groupcomm Profiles" IANA registry.

* Revised the future-ready generalization of 'sign_info_entry'.

* Revised and fixed IANA considerations.

* Fixes and editorial improvements.

C.3. Version -15 to -16

* Distinction between authentication credentials and public keys.

* Consistent renaming of parameters and URI paths.

* Updated format of scope entries when using AIF.

* Updated signaling of semantics for binary encoded scopes.

* Editorial fixes.

C.4. Version -14 to -15

* Fixed nits.

C.5. Version -13 to -14

* Clarified scope and goal of the document in abstract and introduction.

* Overall clarifications on semantics of operations and parameters.

* Major restructuring in the presentation of the KDC interface.

* Revised error handling, also removing redundant text.

* Imported parameters and KDC resource about the KDC's public key from draft-ietf-ace-key-groupcomm-oscore.

* New parameters 'group_rekeying_scheme' and 'control_group_uri'.

* Provided example of administrative keying material transported in 'mgt_key_material'.

* Reasoned categorization of parameters, as expected support by ACE Clients.

* Reasoned categorization of KDC functionalities, as minimally/ optional to support for ACE Clients.

* Guidelines on enhanced error responses using 'error' and 'error_description'.

* New section on group rekeying, discussing at a high-level a basic one-to-one approach and possible one-to-many approaches.

* Revised and expanded security considerations, also about the KDC.

* Updated list of requirements for application profiles.

* Several further clarifications and editorial improvements.

C.6. Version -05 to -13

* Incremental revision of the KDC interface.

* Removed redundancy in parameters about signature algorithm and signature keys.

* Node identifiers always indicated with 'peer_identifiers'.

* Format of public keys changed from raw COSE Keys to be certificates, CWTs or CWT Claims Set (CCS). Adapted parameter 'pub_key_enc'.

* Parameters and functionalities imported from draft-ietf-key-groupcomm-oscore where early defined.

* Possible provisioning of the KDC's Diffie-Hellman public key in response to the Token transferring to /authz-info.

* Generalized proof-of-possession evidence, to be not necessarily a signature.

* Public keys of group members may be retrieved filtering by role and/or node identifier.

* Enhanced error handling with error code and error description.

* Extended "typed" format for the 'scope' claim, optional to use.

* Editorial improvements.

C.7. Version -04 to -05

* Updated uppercase/lowercase URI segments for KDC resources.

* Supporting single Access Token for multiple groups/topics.

* Added 'control_uri' parameter in the Join Request.

* Added 'peer_roles' parameter to support legal requesters/ responders.

* Clarification on stopping using owned keying material.

* Clarification on different reasons for processing failures, related policies, and requirement OPT11.

* Added a KDC sub-resource for group members to upload a new public key.

* Possible group rekeying following an individual Key Renewal Request.

* Clarified meaning of requirement REQ3; added requirement OPT12.

* Editorial improvements.

C.8. Version -03 to -04

* Revised RESTful interface, as to methods and parameters.

* Extended processing of Join Request, as to check/retrieval of public keys.

* Revised and extended profile requirements.

* Clarified specific usage of parameters related to signature algorithms/keys.

* Included general content previously in draft-ietf-ace-key-groupcomm-oscore

* Registration of media type and content format application/ace-group+cbor

* Editorial improvements.

C.9. Version -02 to -03

* Exchange of information on the signature algorithm and related parameters, during the Token POST (Section 3.3).

* Restructured KDC interface, with new possible operations (Section 4).

* Client PoP signature for the Join Request upon joining (Section 4.1.2.1).

* Revised text on group member removal (Section 5).

* Added more profile requirements (Appendix A).

C.10. Version -01 to -02

* Editorial fixes.

* Distinction between transport profile and application profile (Section 1.1).

* New parameters 'sign_info' and 'pub_key_enc' to negotiate parameter values for signature algorithm and signature keys (Section 3.3).

* New parameter 'type' to distinguish different Key Distribution Request messages (Section 4.1).

* New parameter 'client_cred_verify' in the Key Distribution Request to convey a Client signature (Section 4.1).

* Encoding of 'pub_keys_repos' (Section 4.1).

* Encoding of 'mgt_key_material' (Section 4.1).

* Improved description on retrieval of new or updated keying material (Section 6).

* Encoding of 'get_pub_keys' in Public Key Request (Section 7.1).

* Extended security considerations (Sections 10.1 and 10.2).

* New "ACE Public Key Encoding" IANA registry (Section 11.2).

* New "ACE Groupcomm Parameters" IANA registry (Section 11.3), populated with the entries in Section 8.

* New "Ace Groupcomm Request Type" IANA registry (Section 11.4), populated with the values in Section 9.

* New "ACE Groupcomm Policy" IANA registry (Section 11.7) populated with two entries "Sequence Number Synchronization Method" and "Key Update Check Interval" (Section 4.2).

* Improved list of requirements for application profiles (Appendix A).

C.11. Version -00 to -01

* Changed name of 'req_aud' to 'audience' in the Authorization Request (Section 3.1).

* Defined error handling on the KDC (Sections 4.2 and 6.2).

* Updated format of the Key Distribution Response as a whole (Section 4.2).

* Generalized format of 'pub_keys' in the Key Distribution Response (Section 4.2).

* Defined format for the message to request leaving the group (Section 5.2).

* Renewal of individual keying material and methods for group rekeying initiated by the KDC (Section 6).

* CBOR type for node identifiers in 'get_pub_keys' (Section 7.1).

* Added section on parameter identifiers and their CBOR keys (Section 8).

* Added request types for requests to a Join Response (Section 9).

* Extended security considerations (Section 10).

* New IANA registries "ACE Groupcomm Key registry", "ACE Groupcomm Profile registry", "ACE Groupcomm Policy registry" and "Sequence Number Synchronization Method registry" (Section 11).

* Added appendix about requirements for application profiles of ACE on group communication (Appendix A).

Acknowledgments

The following individuals were helpful in shaping this document: Christian Amsüss, Carsten Bormann, Roman Danyliw, Martin Duke, Thomas Fossati, Vidhi Goel, Rikard Höglund, Ben Kaduk, Erik Kline, Warren Kumari, Watson Ladd, John Preuß Mattsson, Daniel Migault, Zaheduzzaman Sarker, Jim Schaad, Ludwig Seitz, Göran Selander, Cigdem Sengul, Dave Thaler, Henry Thompson, Peter van der Stok, and Paul Wouters.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; by the H2020 project SIFIS-Home (Grant agreement 952652); and by the EIT-Digital High Impact Initiative ACTIVE.

Authors' Addresses

Francesca Palombini
Ericsson AB
Torshamnsgatan 23
SE-16440 Stockholm Kista
Sweden
Email: francesca.palombini@ericsson.com

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

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 7 September 2023

M. Tiloca
RISE AB
J. Park
Universitaet Duisburg-Essen
F. Palombini
Ericsson AB

6 March 2023

Key Management for OSCORE Groups in ACE
draft-ietf-ace-key-groupcomm-oscore-16

Abstract

This document defines an application profile of the ACE framework for Authentication and Authorization, to request and provision keying material in group communication scenarios that are based on CoAP and are secured with Group Object Security for Constrained RESTful Environments (Group OSCORE). This application profile delegates the authentication and authorization of Clients, that join an OSCORE group through a Resource Server acting as Group Manager for that group. This application profile leverages protocol-specific transport profiles of ACE to achieve communication security, server authentication and proof-of-possession for a key owned by the Client and bound to an OAuth 2.0 Access Token.

Status of This Memo

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

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

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

This Internet-Draft will expire on 7 September 2023.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 4
1.1. Terminology 5
2. Protocol Overview 6
3. Format of Scope 8
4. Authentication Credentials 10
5. Authorization to Join a Group 12
5.1. Authorization Request 12
5.2. Authorization Response 13
5.3. Token Transferring 14
5.3.1. 'ecdh_info' Parameter 16
5.3.2. 'kdc_dh_creds' Parameter 19
6. Group Joining 20
6.1. Send the Join Request 21
6.1.1. Value of the N_S Challenge 22
6.2. Receive the Join Request 23
6.2.1. Follow-up to a 4.00 (Bad Request) Error Response . . 26
6.3. Send the Join Response 26
6.4. Receive the Join Response 33
7. Overview of the Group Rekeying Process 35
7.1. Stale OSCORE Sender IDs 36
8. Interface at the Group Manager 37
8.1. ace-group/GROUPNAME/active 38
8.1.1. GET Handler 38
8.2. ace-group/GROUPNAME/verif-data 38
8.2.1. GET Handler 38
8.3. ace-group/GROUPNAME/stale-sids 39
8.3.1. FETCH Handler 39
8.4. Admitted Methods 40
8.4.1. Signature Verifiers 41
8.5. Operations Supported by Clients 41
9. Additional Interactions with the Group Manager 42
9.1. Retrieve Updated Keying Material 42
9.1.1. Get Group Keying Material 42
9.1.2. Get Group Keying Material and OSCORE Sender ID . . . 43
9.2. Request to Change Individual Keying Material 43
9.3. Retrieve Authentication Credentials of Group Members . . 45
9.4. Upload a New Authentication Credential 46

9.5. Retrieve the Group Manager's Authentication Credential . 47
9.6. Retrieve Signature Verification Data 48
9.7. Retrieve the Group Policies 50
9.8. Retrieve the Keying Material Version 51
9.9. Retrieve the Group Status 51
9.10. Retrieve Group Names 52
9.11. Leave the Group 55
10. Removal of a Group Member 55
11. Group Rekeying Process 57
11.1. Sending Rekeying Messages 59
11.2. Receiving Rekeying Messages 61
11.3. Missed Rekeying Instances 62
11.3.1. Retrieve Stale Sender IDs 64
12. ACE Groupcomm Parameters 66
13. ACE Groupcomm Error Identifiers 68
14. Default Values for Group Configuration Parameters 69
14.1. Common 70
14.2. Group Mode 70
14.3. Pairwise Mode 71
15. Security Considerations 72
15.1. Management of OSCORE Groups 72
15.2. Size of Nonces as Proof-of-Possesion Challenge 73
15.3. Reusage of Nonces for Proof-of-Possession Input 74
16. IANA Considerations 75
16.1. OAuth Parameters 75
16.2. OAuth Parameters CBOR Mappings 75
16.3. ACE Groupcomm Parameters 76
16.4. ACE Groupcomm Key Types 77
16.5. ACE Groupcomm Profiles 77
16.6. OSCORE Security Context Parameters 78
16.7. TLS Exporter Labels 80
16.8. AIF Media-Type Sub-Parameters 80
16.9. CoAP Content-Format 80
16.10. Group OSCORE Roles 81
16.11. CoRE Resource Type 82
16.12. ACE Groupcomm Errors 82
16.13. Expert Review Instructions 83
17. References 83
17.1. Normative References 83
17.2. Informative References 87
Appendix A. Profile Requirements 88
A.1. Mandatory-to-Address Requirements 89
A.2. Optional-to-Address Requirements 91
Appendix B. Extensibility for Future COSE Algorithms 93
B.1. Format of 'ecdh_info_entry' 94
B.2. Format of 'key' 95
Appendix C. Document Updates 96
C.1. Version -15 to -16 96

C.2. Version -14 to -15 96
C.3. Version -13 to -14 96
C.4. Version -12 to -13 96
C.5. Version -11 to -12 97
C.6. Version -10 to -11 97
C.7. Version -09 to -10 98
C.8. Version -08 to -09 99
C.9. Version -07 to -08 100
C.10. Version -06 to -07 100
C.11. Version -05 to -06 100
C.12. Version -04 to -05 101
C.13. Version -03 to -04 101
C.14. Version -02 to -03 102
C.15. Version -01 to -02 102
C.16. Version -00 to -01 103
Acknowledgments 104
Authors' Addresses 104

1. Introduction

Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] is a method for application-layer protection of the Constrained Application Protocol (CoAP) [RFC7252], using CBOR Object Signing and Encryption (COSE) [RFC9052][RFC9053] and enabling end-to-end security of CoAP payload and options.

As described in [I-D.ietf-core-oscore-groupcomm], Group OSCORE is used to protect CoAP group communication [I-D.ietf-core-groupcomm-bis], which can employ, for example, IP multicast as underlying data transport. This relies on a Group Manager, which is responsible for managing an OSCORE group and enables the group members to exchange CoAP messages secured with Group OSCORE. The Group Manager can be responsible for multiple groups, coordinates the joining process of new group members, and is entrusted with the distribution and renewal of group keying material.

This document is an application profile of [I-D.ietf-ace-key-groupcomm], which itself builds on the ACE framework for Authentication and Authorization [RFC9200]. Message exchanges among the participants as well as message formats and processing follow what specified in [I-D.ietf-ace-key-groupcomm] for provisioning and renewing keying material in group communication scenarios, where Group OSCORE is used to protect CoAP group communication.

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] and in the Authorization Information Format (AIF) [RFC9237] to express authorization information. The terminology for entities in the considered architecture is defined in OAuth 2.0 [RFC6749]. This includes Client (C), Resource Server (RS), and Authorization Server (AS).

* The terms and concepts related to the message formats and processing specified in [I-D.ietf-ace-key-groupcomm], for provisioning and renewing keying material in group communication scenarios. These include the abbreviations REQx and OPTx denoting the numbered mandatory-to-address and optional-to-address requirements, respectively.

* The terms and concepts described in CBOR [RFC8949] and COSE [RFC9052][RFC9053].

* The terms and concepts described in CoAP [RFC7252] and group communication for CoAP [I-D.ietf-core-groupcomm-bis]. Unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol".

* The terms and concepts for protection and processing of CoAP messages through OSCORE [RFC8613] and through Group OSCORE [I-D.ietf-core-oscore-groupcomm] in group communication scenarios. These especially include:

  - Group Manager, as the entity responsible for a set of groups where communications are secured with Group OSCORE. In this document, the Group Manager acts as Resource Server.

  - Authentication credential, as the set of information associated with an entity, including that entity's public key and parameters associated with the public key. Examples of

authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC7925] and C509 certificates [I-D.ietf-cose-cbor-encoded-cert].

Additionally, this document makes use of the following terminology.

* Requester: member of an OSCORE group that sends request messages to other members of the group.

* Responder: member of an OSCORE group that receives request messages from other members of the group. A responder may reply back, by sending a response message to the requester which has sent the request message.

* Monitor: member of an OSCORE group that is configured as responder and never replies back to requesters after receiving request messages. This corresponds to the term "silent server" used in [I-D.ietf-core-oscore-groupcomm].

* Signature verifier: entity external to the OSCORE group and intended to verify the signature of messages exchanged in the group (see Sections 3.1 and 8.5 of [I-D.ietf-core-oscore-groupcomm]). An authorized signature verifier does not join the OSCORE group as an actual member, yet it can retrieve the authentication credentials of the current group members from the Group Manager.

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

* Pairwise-only group: an OSCORE group that uses only the pairwise mode (see Section 9 of [I-D.ietf-core-oscore-groupcomm]).

Examples throughout this document are expressed in CBOR diagnostic notation without the tag and value abbreviations.

2. Protocol Overview

Group communication for CoAP has been enabled in [I-D.ietf-core-groupcomm-bis] and can be secured with Group Object Security for Constrained RESTful Environments (Group OSCORE) as specified in [I-D.ietf-core-oscore-groupcomm]. A network node joins an OSCORE group by interacting with the responsible Group Manager. Once registered in the group, the new node can securely exchange messages with other group members.

This document describes how to use [I-D.ietf-ace-key-groupcomm] and [RFC9200] to perform a number of authentication, authorization and key distribution actions as overviewed in Section 2 of [I-D.ietf-ace-key-groupcomm], when the considered group is specifically an OSCORE group.

With reference to [I-D.ietf-ace-key-groupcomm]:

* The node wishing to join the OSCORE group, i.e., the joining node, is the Client.

* The Group Manager is the Key Distribution Center (KDC), acting as a Resource Server.

* The Authorization Server associated with the Group Manager is the AS.

A node performs the steps described in Sections 3 and 4.3.1.1 of [I-D.ietf-ace-key-groupcomm] in order to obtain an authorization for joining an OSCORE group and then to join that group. The format and processing of messages exchanged during such steps are further specified in Section 5 and Section 6 of this document.

All communications between the involved entities MUST be secured.

In particular, communications between the Client and the Group Manager leverage protocol-specific transport profiles of ACE to achieve communication security, proof-of-possession and server authentication. It is expected that, in the commonly referred base-case of this document, the transport profile to use is pre-configured and well-known to nodes participating in constrained applications.

With respect to what is defined in [I-D.ietf-ace-key-groupcomm]:

* The interface provided by the Group Manager extends the original interface defined in Section 4.1 of [I-D.ietf-ace-key-groupcomm] for the KDC, as specified in Section 8 of this document.

* In addition to those defined in Section 8 of [I-D.ietf-ace-key-groupcomm], additional parameters are defined in this document and summarized in Section 12.

* In addition to those defined in Section 9 of [I-D.ietf-ace-key-groupcomm], additional error identifiers are defined in this document and summarized in Section 13.

Finally, Appendix A lists the specifications on this application profile of ACE, based on the requirements defined in Appendix A of [I-D.ietf-ace-key-groupcomm].

3. Format of Scope

Building on Section 3.1 of [I-D.ietf-ace-key-groupcomm], this section defines the exact format and encoding of scope used in this profile.

To this end, this profile uses the Authorization Information Format (AIF) [RFC9237]. With reference to the generic AIF model

```
AIF-Generic<Toid, Tperm> = [* [Toid, Tperm]]
```

the value of the CBOR byte string used as scope encodes the CBOR array [* [Toid, Tperm]], where each [Toid, Tperm] element corresponds to one scope entry.

Furthermore, this document defines the new AIF specific data model AIF-OSCORE-GROUPCOMM, that this profile MUST use to format and encode scope entries.

In particular, the following holds for each scope entry.

* The object identifier ("Toid") is specialized as a CBOR item specifying the name of the groups pertaining to the scope entry.

* The permission set ("Tperm") is specialized as a CBOR unsigned integer with value R, specifying the permissions that the Client wishes to have in the groups indicated by "Toid".

More specifically, the following applies when, as defined in this document, a scope entry includes as set of permissions the set of roles to take in an OSCORE group.

* The object identifier ("Toid") is a CBOR text string, specifying the group name for the scope entry.

* The permission set ("Tperm") is a CBOR unsigned integer with value R, specifying the role(s) that the Client wishes to take in the group (REQ1). The value R is computed as follows.

  - Each role in the permission set is converted into the corresponding numeric identifier X from the "Value" column of the "Group OSCORE Roles" registry, for which this document defines the entries in Figure 1.

- The set of N numbers is converted into the single value R, by taking two to the power of each numeric identifier X_1, X_2, ..., X_N, and then computing the inclusive OR of the binary representations of all the power values.

Name	Value	Description
Reserved	0	This value is reserved
Requester	1	Send requests; receive responses
Responder	2	Send responses; receive requests
Monitor	3	Receive requests; never send requests/responses
Verifier	4	Verify signature of intercepted messages

Figure 1: Numeric identifier of roles in an OSCORE group

The following CDDL [RFC8610] notation defines a scope entry that uses the AIF-OSCORE-GROUPCOMM data model and expresses a set of Group OSCORE roles from those in Figure 1.

```
AIF-OSCORE-GROUPCOMM = AIF-Generic<oscore-gname, oscore-gperm>

oscore-gname = tstr  ; Group name
oscore-gperm = uint .bits group-oscore-roles

group-oscore-roles = &(
   Requester: 1,
   Responder: 2,
   Monitor: 3,
   Verifier: 4
)

scope_entry = [oscore-gname, oscore-gperm]
```

Future specifications that define new Group OSCORE roles MUST register a corresponding numeric identifier in the "Group OSCORE Roles" registry defined in Section 16.10 of this document.

Note that the value 0 is not available to use as numeric identifier to specify a Group OSCORE role. It follows that, when expressing Group OSCORE roles to take in a group as per this document, a scope entry has the least significant bit of "Tperm" always set to 0.

This is an explicit feature of the AIF-OSCORE-GROUPCOMM data model. That is, for each scope entry, the least significant bit of "Tperm" set to 0 explicitly identifies the scope entry as exactly expressing a set of Group OSCORE roles ("Tperm"), pertaining to a single group whose name is specified by the string literal in "Toid".

Instead, by relying on the same AIF-OSCORE-GROUPCOMM data model, [I-D.ietf-ace-oscore-gm-admin] defines the format of scope entries for Administrator Clients that wish to access an admin interface at the Group Manager. In such scope entries, the least significant bit of "Tperm" is always set to 1.

4. Authentication Credentials

Source authentication of a message sent within the group and protected with Group OSCORE is ensured by means of a digital signature embedded in the message (in group mode), or by integrity-protecting the message with pairwise keying material derived from the asymmetric keys of sender and recipient (in pairwise mode).

Therefore, group members must be able to retrieve each other's authentication credential from a trusted repository, in order to verify source authenticity of incoming group messages.

As also discussed in [I-D.ietf-core-oscore-groupcomm], the Group Manager acts as trusted repository of the authentication credentials of the group members, and provides those authentication credentials to group members if requested to. Upon joining an OSCORE group, a joining node is thus expected to provide its own authentication credential to the Group Manager.

In particular, the following applies when a node joins an OSCORE group.

* The joining node is going to join the group exclusively as monitor, i.e., it is not going to send messages to the group. In this case, the joining node is not required to provide its own authentication credential to the Group Manager, which thus does not have to perform any check related to the format of the authentication credential, to a signature or ECDH algorithm, and to possible parameters associated with the algorithm and the public key. In case the joining node still provides an authentication credential in the 'client_cred' parameter of the Join Request (see Section 6.1), the Group Manager silently ignores that parameter, as well as the related parameters 'cnonce' and 'client_cred_verify'.

* The Group Manager already acquired the authentication credential of the joining node during a past joining process. In this case, the joining node MAY choose not to provide again its own authentication credential to the Group Manager, in order to limit the size of the Join Request. The joining node MUST provide its own authentication credential again if it has provided the Group Manager with multiple authentication credentials during past joining processes, intended for different OSCORE groups. If the joining node provides its own authentication credential, the Group Manager performs consistency checks as per Section 6.2 and, in case of success, considers it as the authentication credential associated with the joining node in the OSCORE group.

* The joining node and the Group Manager use an asymmetric proof-of-possession key to establish a secure communication association. Then, two cases can occur.

  1. When establishing the secure communication association, the Group Manager obtained from the joining node the joining node's authentication credential, in the format used in the OSCORE group and including the asymmetric proof-of-possession key as public key. Also, such authentication credential and the proof-of-possession key are compatible with the signature or ECDH algorithm, and possible associated parameters used in the OSCORE group.

     In this case, the Group Manager considers the authentication credential as the one associated with the joining node in the OSCORE group. If the joining node is aware that the authentication credential and the public key included thereof are also valid for the OSCORE group, then the joining node MAY choose to not provide again its own authentication credential to the Group Manager.

     The joining node MUST provide again its own authentication credential if it has provided the Group Manager with multiple authentication credentials during past joining processes, intended for different OSCORE groups. If the joining node provides its own authentication credential in the 'client_cred' parameter of the Join Request (see Section 6.1), the Group Manager performs consistency checks as per Section 6.2 and, in case of success, considers it as the authentication credential associated with the joining node in the OSCORE group.

2. The authentication credential is not in the format used in the OSCORE group, or else the authentication credential and the proof-of-possession key included as public key are not compatible with the signature or ECDH algorithm, and possible associated parameters used in the OSCORE group.

   In this case, the joining node MUST provide a different compatible authentication credential and public key included thereof to the Group Manager in the 'client_cred' parameter of the Join Request (see Section 6.1). Then, the Group Manager performs consistency checks on this latest provided authentication credential as per Section 6.2 and, in case of success, considers it as the authentication credential associated with the joining node in the OSCORE group.

* The joining node and the Group Manager use a symmetric proof-of-possession key to establish a secure communication association. In this case, upon performing a joining process with that Group Manager for the first time, the joining node specifies its own authentication credential in the 'client_cred' parameter of the Join Request (see Section 6.1).

5. Authorization to Join a Group

This section builds on Section 3 of [I-D.ietf-ace-key-groupcomm] and is organized as follows.

First, Section 5.1 and Section 5.2 describe how the joining node interacts with the AS, in order to be authorized to join an OSCORE group under a given Group Manager and to obtain an Access Token. Then, Section 5.3 describes how the joining node transfers the obtained Access Token to the Group Manager. The following considers a joining node that intends to contact the Group Manager for the first time.

Note that what is defined in Section 3 of [I-D.ietf-ace-key-groupcomm] applies, and only additions or modifications to that specification are defined in this document.

5.1. Authorization Request

The Authorization Request message is as defined in Section 3.1 of [I-D.ietf-ace-key-groupcomm], with the following additions.

* If the 'scope' parameter is present:

- The value of the CBOR byte string encodes a CBOR array, whose format MUST follow the data model AIF-OSCORE-GROUPCOMM defined in Section 3. For each OSCORE group to join:

  - The group name is encoded as a CBOR text string.

  - The set of requested roles is expressed as a single CBOR unsigned integer. This is computed as defined in Section 3, from the numerical abbreviations of each requested role defined in the "Group OSCORE Roles" registry, for which this document defines the entries in Figure 1 (REQ1).

5.2. Authorization Response

The Authorization Response message is as defined in Section 3.2 of [I-D.ietf-ace-key-groupcomm], with the following additions:

* The AS MUST include the 'expires_in' parameter. Other means for the AS to specify the lifetime of Access Tokens are out of the scope of this document.

* The AS MUST include the 'scope' parameter, when the value included in the Access Token differs from the one specified by the joining node in the Authorization Request. In such a case, the second element of each scope entry MUST be present, and specifies the set of roles that the joining node is actually authorized to take in the OSCORE group for that scope entry, encoded as specified in Section 5.1.

Furthermore, the AS MAY use the extended format of scope defined in Section 7 of [I-D.ietf-ace-key-groupcomm] for the 'scope' claim of the Access Token. In such a case, the AS MUST use the CBOR tag with tag number TAG_NUMBER, associated with the CoAP Content-Format CF_ID for the media type application/aif+cbor registered in Section 16.9 of this document (REQ28).

Note to RFC Editor: In the previous paragraph, please replace "TAG_NUMBER" with the CBOR tag number computed as TN(ct) in Section 4.3 of [RFC9277], where ct is the ID assigned to the CoAP Content-Format registered in Section 16.9 of this document. Then, please replace "CF_ID" with the ID assigned to that CoAP Content-Format. Finally, please delete this paragraph.

This indicates that the binary encoded scope, as conveying the actual access control information, follows the scope semantics defined for this application profile in Section 3 of this document.

5.3. Token Transferring

The exchange of Token Transfer Request and Token Transfer Response is defined in Section 3.3 of [I-D.ietf-ace-key-groupcomm]. In addition to that, the following applies.

* The Token Transfer Request MAY additionally contain the following parameters, which, if included, MUST have the corresponding values defined below (OPT2):

  - 'ecdh_info' defined in Section 5.3.1 of this document, with value the CBOR simple value "null" (0xf6) to request information about the ECDH algorithm, the ECDH algorithm parameters, the ECDH key parameters and the exact format of authentication credentials used in the groups that the Client has been authorized to join. This is relevant in case the joining node supports the pairwise mode of Group OSCORE [I-D.ietf-core-oscore-groupcomm].

  - 'kdc_dh_creds' defined in Section 5.3.2 of this document, with value the CBOR simple value "null" (0xf6) to request the Diffie-Hellman authentication credentials of the Group Manager for the groups that the Client has been authorized to join. That is, each of such authentication credentials includes a Diffie-Hellman public key of the Group Manager. This is relevant in case the joining node supports the pairwise mode of Group OSCORE [I-D.ietf-core-oscore-groupcomm].

  Alternatively, the joining node may retrieve this information by other means.

* The 'kdcchallenge' parameter contains a dedicated nonce N_S generated by the Group Manager. For the N_S value, it is RECOMMENDED to use an 8-byte long random nonce. The joining node can use this nonce in order to prove the possession of its own private key, upon joining the group (see Section 6.1).

  The 'kdcchallenge' parameter MAY be omitted from the Token Transfer Response, if the 'scope' of the Access Token specifies only the role "monitor" or only the role "verifier" or only the two roles combined, for each and every of the specified groups.

* If the 'sign_info' parameter is present in the response, the following applies for each element 'sign_info_entry'.

  - 'id' MUST NOT refer to OSCORE groups that are pairwise-only groups.

- 'sign_alg' takes value from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].

- 'sign_parameters' has the same format and value of the COSE capabilities array for the algorithm indicated in 'sign_alg', as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms] (REQ4).

- 'sign_key_parameters' has the same format and value of the COSE capabilities array for the COSE key type of the keys used with the algorithm indicated in 'sign_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types] (REQ5).

- 'cred_fmt' takes value from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters] (REQ6). Consistently with Section 2.3 of [I-D.ietf-core-oscore-groupcomm], acceptable values denote a format of authentication credential that MUST explicitly provide the public key as well as the comprehensive set of information related to the public key algorithm, including, e.g., the used elliptic curve (when applicable).

  At the time of writing this specification, acceptable formats of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC7925] and C509 certificates [I-D.ietf-cose-cbor-encoded-cert]. Further formats may be available in the future, and would be acceptable to use as long as they comply with the criteria defined above.

  [As to CWTs and CCSs, the COSE Header Parameters 'kcwt' and 'kccs' are under pending registration requested by draft-ietf-lake-edhoc.]

  [As to C509 certificates, the COSE Header Parameters 'c5b' and 'c5c' are under pending registration requested by draft-ietf-cose-cbor-encoded-cert.]

This format is consistent with every signature algorithm currently considered in [RFC9053], i.e., with algorithms that have only the COSE key type as their COSE capability. Appendix B of [I-D.ietf-ace-key-groupcomm] describes how the format of each 'sign_info_entry' can be generalized for possible future registered algorithms having a different set of COSE capabilities.

* If 'ecdh_info' is included in the Token Transfer Request, the Group Manager SHOULD include the 'ecdh_info' parameter in the Token Transfer Response, as per the format defined in Section 5.3.1. Note that the field 'id' of each 'ecdh_info_entry' specifies the name, or array of group names, for which that 'ecdh_info_entry' applies to.

  As an exception, the KDC MAY omit the 'ecdh_info' parameter in the Token Transfer Response even if 'ecdh_info' is included in the Token Transfer Request, in case all the groups that the Client is authorized to join are signature-only groups.

* If 'kdc_dh_creds' is included in the Token Transfer Request and any of the groups that the Client has been authorized to join is a pairwise-only group, then the Group Manager MUST include the 'kdc_dh_creds' parameter in the Token Transfer Response, as per the format defined in Section 5.3.2. Otherwise, if 'kdc_dh_creds' is included in the Token Transfer Request, the Group Manager MAY include the 'kdc_dh_creds' parameter in the Token Transfer Response. Note that the field 'id' specifies the group name, or array of group names, for which the corresponding 'kdc_dh_creds' applies to.

Note that, other than through the above parameters as defined in Section 3.3 of [I-D.ietf-ace-key-groupcomm], the joining node may have obtained such information by alternative means. For example, information conveyed in the 'sign_info' and 'ecdh_info' parameters may have been pre-configured, or the joining node may early retrieve it, e.g., by using the approach described in [I-D.tiloca-core-oscore-discovery] to discover the OSCORE group and the link to the associated group-membership resource at the Group Manager (OPT3).

5.3.1. 'ecdh_info' Parameter

The 'ecdh_info' parameter is an OPTIONAL parameter of the request and response messages exchanged between the Client and the authz-info endpoint at the RS (see Section 5.10.1. of [RFC9200]).

This parameter allows the Client and the RS to exchange information about an ECDH algorithm as well as about the authentication credentials and public keys to accordingly use for deriving Diffie-Hellman secrets. Its exact semantics and content are application specific.

In this application profile, this parameter is used to exchange information about the ECDH algorithm as well as about the authentication credentials and public keys to be used with it, in the groups indicated by the transferred Access Token as per its 'scope' claim (see Section 3.2 of [I-D.ietf-ace-key-groupcomm]).

When used in the Token Transfer Request sent to the Group Manager, the 'ecdh_info' parameter has value the CBOR simple value "null" (0xf6). This is done to ask for information about the ECDH algorithm as well as about the authentication credentials and public keys to be used to compute static-static Diffie-Hellman shared secrets [NIST-800-56A], in the OSCORE groups that the Client has been authorized to join and that use the pairwise mode of Group OSCORE [I-D.ietf-core-oscore-groupcomm].

When used in the following Token Transfer Response from the Group Manager, the 'ecdh_info' parameter is a CBOR array of one or more elements. The number of elements is at most the number of OSCORE groups that the Client has been authorized to join.

Each element contains information about ECDH parameters as well as about authentication credentials and public keys, for one or more OSCORE groups that use the pairwise mode of Group OSCORE and that the Client has been authorized to join. Each element is formatted as follows.

* The first element 'id' is the group name of the OSCORE group or an array of group names for the OSCORE groups for which the specified information applies. In the following, each specified group name is referred to as 'gname'. The 'id' element MUST NOT refer to OSCORE groups that are signature-only groups.

* The second element 'ecdh_alg' is a CBOR integer or a CBOR text string indicating the ECDH algorithm used in the OSCORE group identified by 'gname'. Values are taken from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].

* The third element 'ecdh_parameters' is a CBOR array indicating the parameters of the ECDH algorithm used in the OSCORE group identified by 'gname'. Its format and value are the same of the COSE capabilities array for the algorithm indicated in 'ecdh_alg', as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms].

* The fourth element 'ecdh_key_parameters' is a CBOR array indicating the parameters of the keys used with the ECDH algorithm in the OSCORE group identified by 'gname'. Its content depends on the value of 'ecdh_alg'. In particular, its format and value are

the same of the COSE capabilities array for the COSE key type of the keys used with the algorithm indicated in 'ecdh_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types].

* The fifth element 'cred_fmt' is a CBOR integer indicating the format of authentication credentials used in the OSCORE group identified by 'gname'. It takes value from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters] (REQ6). Acceptable values denote a format that MUST explicitly provide the public key as well as a comprehensive set of information related to the public key algorithm. This information includes, e.g., the used elliptic curve (when applicable). The same considerations and guidelines for the 'cred_fmt' element of 'sign_info' apply (see Section 5.3).

The CDDL notation [RFC8610] of the 'ecdh_info' parameter is given below.

```
ecdh_info = ecdh_info_req / ecdh_info_resp

ecdh_info_req = null                    ; in the Token Transfer
                                        ; Request to the
                                        ; Group Manager

ecdh_info_res = [ + ecdh_info_entry ] ; in the Token Transfer
                                        ; Response from the
                                        ; Group Manager

ecdh_info_entry =
[
  id : gname / [ + gname ],
  ecdh_alg : int / tstr,
  ecdh_parameters : [ any ],
  ecdh_key_parameters : [ any ],
  cred_fmt : int
]

gname = tstr
```

This format is consistent with every ECDH algorithm currently defined in [RFC9053], i.e., with algorithms that have only the COSE key type as their COSE capability. Appendix B of this document describes how the format of each 'ecdh_info_entry' can be generalized for possible future registered algorithms having a different set of COSE capabilities.

5.3.2. 'kdc_dh_creds' Parameter

The 'kdc_dh_creds' parameter is an OPTIONAL parameter of the request and response messages exchanged between the Client and the authz-info endpoint at the RS (see Section 5.10.1. of [RFC9200]).

This parameter allows the Client to request and retrieve the Diffie-Hellman authentication credentials of the RS, i.e., authentication credentials including a Diffie-Hellman public key of the RS.

In this application profile, this parameter is used to request and retrieve from the Group Manager its Diffie-Hellman authentication credentials to use, in the OSCORE groups that the Client has been authorized to join. The Group Manager has a Diffie-Hellman authentication credential in an OSCORE group if and only if the group is a pairwise-only group. In this case, the early retrieval of the Group Manager's authentication credential is necessary in order for the joining node to prove the possession of its own private key, upon joining the group (see Section 6.1).

When used in the Token Transfer Request sent to the Group Manager, the 'kdc_dh_creds' parameter has value the CBOR simple value "null" (0xf6). This is done to ask for the Diffie-Hellman authentication credentials that the Group Manager uses in the OSCORE groups that the Client has been authorized to join.

When used in the following Token Transfer Response from the Group Manager, the 'kdc_dh_creds' parameter is a CBOR array of one or more elements. The number of elements is at most the number of OSCORE groups that the Client has been authorized to join.

Each element 'kdc_dh_creds_entry' contains information about the Group Manager's Diffie-Hellman authentication credentials, for one or more OSCORE groups that are pairwise-only groups and that the Client has been authorized to join. Each element is formatted as follows.

* The first element 'id' is the group name of the OSCORE group or an array of group names for the OSCORE groups for which the specified information applies. In particular, 'id' MUST refer exclusively to OSCORE groups that are pairwise-only groups.

* The second element 'cred_fmt' is a CBOR integer indicating the format of authentication credentials used in the OSCORE group identified by 'gname'. It takes value from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters] (REQ6). Acceptable values denote a format that MUST explicitly provide the public key as well as a comprehensive set of information related to the public key algorithm. This information

includes, e.g., the used elliptic curve (when applicable). The same considerations and guidelines for the 'cred_fmt' element of 'sign_info' apply (see Section 5.3).

* The third element 'cred' is a CBOR byte string, which encodes the Group Manager's Diffie-Hellman authentication credential in its original binary representation made available to other endpoints in the group. That is, the original binary representation complies with the format specified by the 'cred_fmt' element. Note that the authentication credential provides the comprehensive set of information related to its public key algorithm, i.e., the ECDH algorithm used in the OSCORE group as pairwise key agreement algorithm.

The CDDL notation [RFC8610] of the 'kdc_dh_creds' parameter is given below.

```
kdc_dh_creds = kdc_dh_creds_req / kdc_dh_creds_resp

kdc_dh_creds_req = null                     ; in the Token Transfer
                                            ; Request to the
                                            ; Group Manager

kdc_dh_creds_res = [ + kdc_dh_creds_entry ] ; in the Token Transfer
                                            ; Response from the
                                            ; Group Manager

kdc_dh_creds_entry =
[
  id : gname / [ + gname ],
  cred_fmt : int,
  cred : bstr
]

gname = tstr
```

6. Group Joining

This section describes the interactions between the joining node and the Group Manager to join an OSCORE group. The message exchange between the joining node and the Group Manager consists of the messages defined in Section 4.3.1.1 of [I-D.ietf-ace-key-groupcomm]. Note that what is defined in [I-D.ietf-ace-key-groupcomm] applies, and only additions or modifications to that specification are defined in this document.

6.1. Send the Join Request

The joining node requests to join the OSCORE group by sending a Join Request message to the related group-membership resource at the Group Manager, as per Section 4.3.1.1 of [I-D.ietf-ace-key-groupcomm]. Additionally to what is defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], the following applies.

* The 'scope' parameter MUST be included. Its value encodes one scope entry with the format defined in Section 3, indicating the group name and the role(s) that the joining node wants to take in the group.

  The 'scope' parameter MUST NOT specify any of the following sets of roles: ("requester", "monitor") and ("responder", "monitor"). Future specifications that define a new role for members of OSCORE groups MUST define possible sets of roles (including the new role and existing roles) that are not acceptable to specify in the 'scope' parameter of a Join Request.

* The 'get_creds' parameter is present only if the joining node wants to retrieve the authentication credentials of the group members from the Group Manager during the joining process (see Section 4). Otherwise, this parameter MUST NOT be present.

  If this parameter is present and its value is not the CBOR simple value "null" (0xf6), each element of the inner CBOR array 'role_filter' is encoded as a CBOR unsigned integer, with the same value of a permission set ("Tperm") indicating that role or combination of roles in a scope entry, as defined in Section 3.

* 'cnonce' contains a dedicated nonce N_C generated by the joining node. For the N_C value, it is RECOMMENDED to use an 8-byte long random nonce.

* The proof-of-possession (PoP) evidence included in 'client_cred_verify' is computed as defined below (REQ14). In either case, the N_S used to build the PoP input is as defined in Section 6.1.1.

  - If the group is not a pairwise-only group, the PoP evidence MUST be a signature. The joining node computes the signature by using the same private key and signature algorithm it intends to use for signing messages in the OSCORE group.

- If the group is a pairwise-only group, the PoP evidence MUST be a MAC computed as follows, by using the HKDF Algorithm HKDF SHA-256, which consists of composing the HKDF-Extract and HKDF-Expand steps [RFC5869].

  MAC = HKDF(salt, IKM, info, L)

  The input parameters of HKDF are as follows.

  - salt takes as value the empty byte string.

  - IKM is computed as a cofactor Diffie-Hellman shared secret, see Section 5.7.1.2 of [NIST-800-56A], using the ECDH algorithm used in the OSCORE group. The joining node uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the Group Manager. For X25519 and X448, the procedure is described in Section 5 of [RFC7748].

  - info takes as value the PoP input.

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

6.1.1. Value of the N_S Challenge

The value of the N_S challenge is determined as follows.

1. If the joining node has provided the Access Token to the Group Manager by means of a Token Transfer Request to the /authz-info endpoint as in Section 5.3, then N_S takes the same value of the most recent 'kdcchallenge' parameter received by the joining node from the Group Manager. This can be either the one specified in the Token Transfer Response, or the one possibly specified in a 4.00 (Bad Request) error response to a following Join Request (see Section 6.2).

2. If the provisioning of the Access Token to the Group Manager has relied on the DTLS profile of ACE [RFC9202], and the Access Token was specified:

   * in the "psk_identity" field of the ClientKeyExchange message when using DTLS 1.2 [RFC6347]; or

   * in the "identity" field of a PskIdentity within the PreSharedKeyExtension of the ClientHello message when using DTLS 1.3 [RFC9147],

then N_S is an exporter value computed as defined in Section 7.5 of [RFC8446]. Specifically, N_S is exported from the DTLS session between the joining node and the Group Manager, using an empty 'context_value', 32 bytes as 'key_length', and the exporter label "EXPORTER-ACE-Sign-Challenge-coap-group-oscore-app" defined in Section 16.7 of this document.

It is up to applications to define how N_S is computed in further alternative settings.

Section 15.3 provides security considerations on the reusage of the N_S challenge.

6.2. Receive the Join Request

The Group Manager processes the Join Request as defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], with the following additions.

The Group Manager verifies the PoP evidence contained in 'client_cred_verify' as follows:

* As PoP input, the Group Manager uses the value of the 'scope' parameter from the Join Request as a CBOR byte string, concatenated with N_S encoded as a CBOR byte string, concatenated with N_C encoded as a CBOR byte string. The value of N_S is determined as described in Section 6.1.1, while N_C is the nonce provided in the 'cnonce' parameter of the Join Request.

* As public key of the joining node, the Group Manager uses either the one included in the authentication credential retrieved from the 'client_cred' parameter of the Join Request, or the one from the already stored authentication credential as acquired from previous interactions with the joining node (see Section 4).

* If the group is not a pairwise-only group, the PoP evidence is a signature. The Group Manager verifies it by using the public key of the joining node, as well as the signature algorithm used in the OSCORE group and possible corresponding parameters.

* If the group is a pairwise-only group, the PoP evidence is a MAC. The Group Manager recomputes the MAC through the same process taken by the joining node when preparing the value of the 'client_cred_verify' parameter for the Join Request (see Section 6.1), with the difference that the Group Manager uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the joining node. The verification succeeds if and only if the recomputed MAC is equal to the MAC conveyed as PoP evidence in the Join Request.

The Group Manager MUST reply with a 5.03 (Service Unavailable) error response in the following cases:

* There are currently no OSCORE Sender IDs available to assign in the OSCORE group and, at the same time, the joining node is not going to join the group exclusively as monitor. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 4 ("No available node identifiers").

* The OSCORE group that the joining node has been trying to join is currently inactive (see Section 8.1). The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 9 ("Group currently not active").

The Group Manager MUST reply with a 4.00 (Bad Request) error response in the following cases:

* The 'client_cred' parameter is present in the Join Request and its value is not an eligible authentication credential (e.g., it is not of the format accepted in the group).

* The 'client_cred' parameter is not present in the Join Request while the joining node is not going to join the group exclusively as monitor, and any of the following conditions holds:

  - The Group Manager does not store an eligible authentication credential (e.g., of the format accepted in the group) for the joining node.

  - The Group Manager stores multiple eligible authentication credentials (e.g., of the format accepted in the group) for the joining node.

* The 'scope' parameter is not present in the Join Request, or it is present and specifies any of the following sets of roles: ("requester", "monitor") and ("responder", "monitor").

* The Join Request includes the 'client_cred' parameter but does not include both the 'cnonce' and 'client_cred_verify' parameters.

In order to prevent the acceptance of Ed25519 and Ed448 public keys that cannot be successfully converted to Montgomery coordinates, and thus cannot be used for the derivation of pairwise keys (see Section 2.4.1 of [I-D.ietf-core-oscore-groupcomm]), the Group Manager MAY reply with a 4.00 (Bad Request) error response in case all the following conditions hold:

* The OSCORE group uses the pairwise mode of Group OSCORE.

* The OSCORE group uses EdDSA public keys [RFC8032].

* The authentication credential of the joining node from the 'client_cred' parameter includes a public key which:

  - Is for the elliptic curve Ed25519 and has its Y coordinate equal to -1 or 1 (mod p), with p = (2^255 - 19), see Section 4.1 of [RFC7748]; or

  - Is for the elliptic curve Ed448 and has its Y coordinate equal to -1 or 1 (mod p), with p = (2^448 - 2^224 - 1), see Section 4.2 of [RFC7748].

A 4.00 (Bad Request) error response from the Group Manager to the joining node MUST have content format application/ace-groupcomm+cbor. The response payload is a CBOR map formatted as follows:

* If the group uses (also) the group mode of Group OSCORE, the CBOR map MUST contain the 'sign_info' parameter, whose CBOR label is defined in Section 8 of [I-D.ietf-ace-key-groupcomm]. This parameter has the same format of 'sign_info_res' defined in Section 3.3.1 of [I-D.ietf-ace-key-groupcomm] and includes a single element 'sign_info_entry', pertaining to the OSCORE group that the joining node has tried to join with the Join Request.

* If the group uses (also) the pairwise mode of Group OSCORE, the CBOR map MUST contain the 'ecdh_info' parameter, whose CBOR label is defined in Section 16.3. This parameter has the same format of 'ecdh_info_res' defined in Section 5.3.1 and includes a single element 'ecdh_info_entry', pertaining to the OSCORE group that the joining node has tried to join with the Join Request.

* If the group is a pairwise-only group, the CBOR map MUST contain the 'kdc_dh_creds' parameter, whose CBOR label is defined in Section 16.3. This parameter has the same format of 'kdc_dh_creds_res' defined in Section 5.3.2 and includes a single element 'kdc_dh_creds_entry', pertaining to the OSCORE group that the joining node has tried to join with the Join Request.

* The CBOR map MAY include the 'kdcchallenge' parameter, whose CBOR label is defined in Section 8 of [I-D.ietf-ace-key-groupcomm]. If present, this parameter is a CBOR byte string, which encodes a newly generated 'kdcchallenge' value that the Client can use when preparing a Join Request (see Section 6.1). In such a case the Group Manager MUST store the newly generated value as the 'kdcchallenge' value associated with the joining node, replacing the currently stored value (if any).

6.2.1. Follow-up to a 4.00 (Bad Request) Error Response

When receiving a 4.00 (Bad Request) error response, the joining node MAY send a new Join Request to the Group Manager. In such a case:

* The 'cnonce' parameter MUST include a new dedicated nonce N_C generated by the joining node.

* The 'client_cred' parameter MUST include an authentication credential in the format indicated by the Group Manager. Also, the authentication credential as well as the included public key MUST be compatible with the signature or ECDH algorithm, and possible associated parameters.

* The 'client_cred_verify' parameter MUST include a PoP evidence computed as described in Section 6.1, by using the private key associated with the authentication credential specified in the current 'client_cred' parameter, with the signature or ECDH algorithm, and possible associated parameters indicated by the Group Manager. If the error response from the Group Manager includes the 'kdcchallenge' parameter, the joining node MUST use its content as new N_S challenge to compute the PoP evidence.

6.3. Send the Join Response

If the processing of the Join Request described in Section 6.2 is successful, the Group Manager updates the group membership by registering the joining node NODENAME as a new member of the OSCORE group GROUPNAME, as described in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm].

If the joining node has not taken exclusively the role of monitor, the Group Manager performs also the following actions.

* The Group Manager selects an available OSCORE Sender ID in the OSCORE group, and exclusively assigns it to the joining node. The Group Manager MUST NOT assign an OSCORE Sender ID to the joining node if this joins the group exclusively with the role of monitor, according to what is specified in the Access Token (see Section 5.2).

  Consistently with Section 3.2.1 of [I-D.ietf-core-oscore-groupcomm], the Group Manager MUST assign an OSCORE Sender ID that has not been used in the OSCORE group since the latest time when the current Gid value was assigned to the group.

  If the joining node is recognized as a current group member, e.g., through the ongoing secure communication association, the following also applies.

  - The Group Manager MUST assign a new OSCORE Sender ID different than the one currently used by the joining node in the OSCORE group.

  - The Group Manager MUST add the old, relinquished OSCORE Sender ID of the joining node to the set of stale Sender IDs associated with the current version of the group keying material for the group (see Section 7.1).

* The Group Manager stores the association between i) the authentication credential of the joining node; and ii) the Group Identifier (Gid), i.e., the OSCORE ID Context, associated with the OSCORE group together with the OSCORE Sender ID assigned to the joining node in the group. The Group Manager MUST keep this association updated over time.

Then, the Group Manager replies to the joining node, providing the updated security parameters and keying material necessary to participate in the group communication. This success Join Response is formatted as defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], with the following additions:

* The 'gkty' parameter identifies a key of type "Group_OSCORE_Input_Material object", defined in Section 16.4 of this document.

* The 'key' parameter includes what the joining node needs in order to set up the Group OSCORE Security Context as per Section 2 of [I-D.ietf-core-oscore-groupcomm].

  This parameter has as value a Group_OSCORE_Input_Material object, which is defined in this document and extends the OSCORE_Input_Material object encoded in CBOR as defined in Section 3.2.1 of [RFC9203]. In particular, it contains the additional parameters 'group_senderId', 'cred_fmt', 'sign_enc_alg', 'sign_alg', 'sign_params', 'ecdh_alg' and 'ecdh_params' defined in Section 16.6 of this document.

  More specifically, the 'key' parameter is composed as follows.

  - The 'hkdf' parameter, if present, specifies the HKDF Algorithm used in the OSCORE group. The HKDF Algorithm is specified by the HMAC Algorithm value. This parameter MAY be omitted, if the HKDF Algorithm used in the group is HKDF SHA-256. Otherwise, this parameter MUST be present.

  - The 'salt' parameter, if present, has as value the OSCORE Master Salt used in the OSCORE group. This parameter MAY be omitted, if the Master Salt used in the group is the empty byte string. Otherwise, this parameter MUST be present.

  - The 'ms' parameter includes the OSCORE Master Secret value used in the OSCORE group. This parameter MUST be present.

  - The 'contextId' parameter has as value the Group Identifier (Gid), i.e., the OSCORE ID Context of the OSCORE group. This parameter MUST be present.

  - The 'group_senderId' parameter has as value the OSCORE Sender ID assigned to the joining node by the Group Manager, as described above. This parameter MUST be present if and only if the node does not join the OSCORE group exclusively with the role of monitor, according to what is specified in the Access Token (see Section 5.2).

- The 'cred_fmt' parameter specifies the format of authentication credentials used in the OSCORE group. This parameter MUST be present and it takes value from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters] (REQ6). Consistently with Section 2.3 of [I-D.ietf-core-oscore-groupcomm], acceptable values denote a format that MUST explicitly provide the public key as well as a comprehensive set of information related to the public key algorithm. This information includes, e.g., the used elliptic curve (when applicable).

  At the time of writing this specification, acceptable formats of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC7925] and C509 certificates [I-D.ietf-cose-cbor-encoded-cert]. Further formats may be available in the future, and would be acceptable to use as long as they comply with the criteria defined above.

  [As to CWTs and CCSs, the COSE Header Parameters 'kcwt' and 'kccs' are under pending registration requested by draft-ietf-lake-edhoc.]

  [As to C509 certificates, the COSE Header Parameters 'c5b' and 'c5c' are under pending registration requested by draft-ietf-cose-cbor-encoded-cert.]

The 'key' parameter MUST also include the following parameters, if and only if the OSCORE group is not a pairwise-only group.

- The 'sign_enc_alg' parameter, specifying the Signature Encryption Algorithm used in the OSCORE group to encrypt messages protected with the group mode. This parameter takes values from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].

- The 'sign_alg' parameter, specifying the Signature Algorithm used to sign messages in the OSCORE group. This parameter takes values from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].

- The 'sign_params' parameter, specifying the parameters of the Signature Algorithm. This parameter is a CBOR array, which includes the following two elements:

- o 'sign_alg_capab': a CBOR array, with the same format and value of the COSE capabilities array for the Signature Algorithm indicated in 'sign_alg', as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms].

- o 'sign_key_type_capab': a CBOR array, with the same format and value of the COSE capabilities array for the COSE key type of the keys used with the Signature Algorithm indicated in 'sign_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types].

The 'key' parameter MUST also include the following parameters, if and only if the OSCORE group is not a signature-only group.

- The 'alg' parameter, specifying the AEAD Algorithm used in the OSCORE group to encrypt messages protected with the pairwise mode.

- The 'ecdh_alg' parameter, specifying the Pairwise Key Agreement Algorithm used in the OSCORE group. This parameter takes values from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].

- The 'ecdh_params' parameter, specifying the parameters of the Pairwise Key Agreement Algorithm. This parameter is a CBOR array, which includes the following two elements:

  - o 'ecdh_alg_capab': a CBOR array, with the same format and value of the COSE capabilities array for the algorithm indicated in 'ecdh_alg', as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms].

  - o 'ecdh_key_type_capab': a CBOR array, with the same format and value of the COSE capabilities array for the COSE key type of the keys used with the algorithm indicated in 'ecdh_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types].

The format of 'key' defined above is consistent with every signature algorithm and ECDH algorithm currently considered in [RFC9053], i.e., with algorithms that have only the COSE key type as their COSE capability. Appendix B of this document describes how the format of the 'key' parameter can be generalized for possible future registered algorithms having a different set of COSE capabilities.

Furthermore, the following applies.

* The 'exp' parameter MUST be present.

* The 'ace_groupcomm_profile' parameter MUST be present and has value coap_group_oscore_app (PROFILE_TBD), which is defined in Section 16.5 of this document.

* The 'creds' parameter, if present, includes the authentication credentials requested by the joining node by means of the 'get_creds' parameter in the Join Request.

  If the joining node has asked for the authentication credentials of all the group members, i.e., 'get_creds' had value the CBOR simple value "null" (0xf6) in the Join Request, then the Group Manager provides only the authentication credentials of the group members that are relevant to the joining node. That is, in such a case, 'creds' includes only: i) the authentication credentials of the responders currently in the OSCORE group, in case the joining node is configured (also) as requester; and ii) the authentication credentials of the requesters currently in the OSCORE group, in case the joining node is configured (also) as responder or monitor.

* The 'peer_identifiers' parameter includes the OSCORE Sender ID of each group member whose authentication credential is specified in the 'creds' parameter. That is, a group member's Sender ID is used as identifier for that group member (REQ25).

* The 'group_policies' parameter SHOULD be present, and SHOULD include the following elements:

  - "Key Update Check Interval" defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], with default value 3600;

  - "Expiration Delta" defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], with default value 0.

* The 'kdc_cred' parameter MUST be present, specifying the Group Manager's authentication credential in its original binary representation (REQ8). The Group Manager's authentication credential MUST be in the format used in the OSCORE group. Also, the authentication credential as well as the included public key MUST be compatible with the signature or ECDH algorithm, and possible associated parameters used in the OSCORE group.

* The 'kdc_nonce' parameter MUST be present, specifying the dedicated nonce N_KDC generated by the Group Manager. For N_KDC, it is RECOMMENDED to use an 8-byte long random nonce.

* The 'kdc_cred_verify' parameter MUST be present, specifying the proof-of-possession (PoP) evidence computed by the Group Manager. The PoP evidence is computed over the nonce N_KDC, which is specified in the 'kdc_nonce' parameter and taken as PoP input. The PoP evidence is computed as defined below (REQ21).

  - If the group is not a pairwise-only group, the PoP evidence MUST be a signature. The Group Manager computes the signature by using the signature algorithm used in the OSCORE group, as well as its own private key associated with the authentication credential specified in the 'kdc_cred' parameter.

  - If the group is a pairwise-only group, the PoP evidence MUST be a MAC computed as follows, by using the HKDF Algorithm HKDF SHA-256, which consists of composing the HKDF-Extract and HKDF-Expand steps [RFC5869].

    MAC = HKDF(salt, IKM, info, L)

    The input parameters of HKDF are as follows.

    o salt takes as value the empty byte string.

    o IKM is computed as a cofactor Diffie-Hellman shared secret, see Section 5.7.1.2 of [NIST-800-56A], using the ECDH algorithm used in the OSCORE group. The Group Manager uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the joining node. For X25519 and X448, the procedure is described in Section 5 of [RFC7748].

    o info takes as value the PoP input.

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

* The 'group_rekeying' parameter MAY be omitted, if the Group Manager uses the "Point-to-Point" group rekeying scheme registered in Section 11.12 of [I-D.ietf-ace-key-groupcomm] as rekeying scheme in the OSCORE group (OPT9). Its detailed use for this profile is defined in Section 11 of this document. In any other case, the 'group_rekeying' parameter MUST be included.

As a last action, if the Group Manager reassigns Gid values during the group's lifetime (see Section 3.2.1.1 of [I-D.ietf-core-oscore-groupcomm]), then the Group Manager MUST store the Gid specified in the 'contextId' parameter of the 'key' parameter, as the Birth Gid of the joining node in the joined group (see Section 3 of [I-D.ietf-core-oscore-groupcomm]). This applies also in case the joining node is in fact re-joining the group; in such a case, the newly determined Birth Gid overwrites the one currently stored.

6.4. Receive the Join Response

Upon receiving the Join Response, the joining node retrieves the Group Manager's authentication credential from the 'kdc_cred' parameter. The joining node MUST verify the proof-of-possession (PoP) evidence specified in the 'kdc_cred_verify' parameter of the Join Response as defined below (REQ21).

* If the group is not a pairwise-only group, the PoP evidence is a signature. The joining node verifies it by using the public key of the Group Manager from the received authentication credential, as well as the signature algorithm used in the OSCORE group and possible corresponding parameters.

* If the group is a pairwise-only group, the PoP evidence is a MAC. The joining node recomputes the MAC through the same process taken by the Group Manager when computing the value of the 'kdc_cred_verify' parameter (see Section 6.3), with the difference that the joining node uses its own Diffie-Hellman private key and the Diffie-Hellman public key of the Group Manager from the received authentication credential. The verification succeeds if and only if the recomputed MAC is equal to the MAC conveyed as PoP evidence in the Join Response.

In case of failed verification of the PoP evidence, the joining node MUST stop processing the Join Response and MAY send a new Join Request to the Group Manager (see Section 6.1).

In case of successful verification of the PoP evidence, the joining node uses the information received in the Join Response to set up the Group OSCORE Security Context, as described in Section 2 of

[I-D.ietf-core-oscore-groupcomm]. If the following parameters were not included in the 'key' parameter of the Join Response, the joining node considers the default values specified below, consistently with Section 3.2 of [RFC8613].

* Absent the 'hkdf' parameter, the joining node considers HKDF SHA-256 as HKDF Algorithm to use in the OSCORE group.

* Absent the 'salt' parameter, the joining node considers the empty byte string as Master Salt to use in the OSCORE group.

* Absent the 'group_rekeying' parameter, the joining node considers the "Point-to-Point" group rekeying scheme registered in Section 11.12 of [I-D.ietf-ace-key-groupcomm] as the rekeying scheme used in the group (OPT9). Its detailed use for this profile is defined in Section 11 of this document.

In addition, the joining node maintains an association between each authentication credential retrieved from the 'creds' parameter and the role(s) that the corresponding group member has in the OSCORE group.

From then on, the joining node can exchange group messages secured with Group OSCORE as described in [I-D.ietf-core-oscore-groupcomm]. When doing so:

* The joining node MUST NOT process an incoming request message, if protected by a group member whose authentication credential is not associated with the role "Requester".

* The joining node MUST NOT process an incoming response message, if protected by a group member whose authentication credential is not associated with the role "Responder".

* The joining node MUST NOT use the pairwise mode of Group OSCORE to process messages in the group, if the Join Response did not include the 'ecdh_alg' parameter.

If the application requires backward security, the Group Manager MUST generate updated security parameters and group keying material, and provide it to the current group members, upon the new node's joining (see Section 11). In such a case, the joining node is not able to access secure communication in the OSCORE group that occurred prior to its joining.

7. Overview of the Group Rekeying Process

In a number of cases, the Group Manager has to generate new keying material and distribute it to the group (rekeying), as also discussed in Section 3.2 of [I-D.ietf-core-oscore-groupcomm].

To this end the Group Manager MUST support the Group Rekeying Process described in Section 11 of this document, as an instance of the "Point-to-Point" rekeying scheme defined in Section 6.1 of [I-D.ietf-ace-key-groupcomm] and registered in Section 11.12 of [I-D.ietf-ace-key-groupcomm]. Future documents may define the use of alternative group rekeying schemes for this application profile, together with the corresponding rekeying message formats. The resulting group rekeying process MUST comply with the functional steps defined in Section 3.2 of [I-D.ietf-core-oscore-groupcomm].

Upon generating the new group keying material and before starting its distribution, the Group Manager MUST increment the version number of the group keying material. When rekeying a group, the Group Manager MUST preserve the current value of the OSCORE Sender ID of each member in that group.

The data distributed to a group through a rekeying MUST include:

* The new version number of the group keying material for the group.

* A new Group Identifier (Gid) for the group as introduced in [I-D.ietf-ace-key-groupcomm], used as ID Context parameter of the Group OSCORE Common Security Context of that group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]).

  Note that the Gid differs from the group name also introduced in [I-D.ietf-ace-key-groupcomm], which is a plain, stable and invariant identifier, with no cryptographic relevance and meaning.

* A new value for the Master Secret parameter of the Group OSCORE Common Security Context of the group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]).

* A set of stale Sender IDs, which allows each rekeyed node to purge authentication credentials and Recipient Contexts used in the group and associated with those Sender IDs. This in turn allows every group member to rely on stored authentication credentials, in order to confidently assert the group membership of other sender nodes, when receiving protected messages in the group (see Section 3.2 of [I-D.ietf-core-oscore-groupcomm]). More details on the maintenance of stale Sender IDs are provided in Section 7.1.

Also, the data distributed through a group rekeying MAY include a new value for the Master Salt parameter of the Group OSCORE Common Security Context of that group.

The Group Manager MUST rekey the group in the following cases.

* The application requires backward security - In this case, the group is rekeyed when a node joins the group as a new member. Therefore, a joining node cannot access communications in the group prior to its joining.

* One or more nodes leave the group - That is, the group is rekeyed when one or more current members spontaneously request to leave the group (see Section 9.11), or when the Group Manager forcibly evicts them from the group, e.g., due to expired or revoked authorization (see Section 10). Therefore, a leaving node cannot access communications in the group after its leaving, thus ensuring forward security in the group.

  Due to the set of stale Sender IDs distributed through the rekeying, this ensures that a node owning the latest group keying material does not store the authentication credentials of former group members (see Sections 3.2 and 12.1 of [I-D.ietf-core-oscore-groupcomm]).

When the expiration time for the group keying material approaches or has passed, the Group Manager may want to extend the secure group operation, as considered appropriate. If the Group Manager does so, the Group Manager MUST rekey the group.

The Group Manager MAY rekey the group for other reasons, e.g., according to an application-specific rekeying period or scheduling.

7.1. Stale OSCORE Sender IDs

For each OSCORE group, the Group Manager MUST maintain N > 1 sets of "stale" OSCORE Sender IDs. It is up to the application to specify the value of N, possibly on a per-group basis.

Each set is uniquely associated with one version of the group keying material, and includes the OSCORE Sender IDs that have become "stale" in the OSCORE group under that version of the group keying material.

In the following cases, the Group Manager MUST add an element to the set X associated with the current version of the group keying material.

* When a current group member obtains a new Sender ID, its old Sender ID is added to X. This happens when the Group Manager assigns a new Sender ID upon request from the group member (see Section 9.2), or in case the group member re-joins the group (see Section 6.1 and Section 6.3), thus also obtaining a new Sender ID.

* When a current group member leaves the group, its current Sender ID is added to X. This happens when a group member requests to leave the group (see Section 9.11) or is forcibly evicted from the group (see Section 10).

The value of N can change during the lifetime of the group. If the new value N' is smaller than N, the Group Manager MUST preserve the sets associated with the (up to) N' most recent versions of the group keying material.

When performing a group rekeying (see Section 11) for switching from an old version V to a new version V' = (V + 1) of the group keying material, the Group Manager MUST perform the following actions.

* Before creating the new group keying material with version V', if the number of sets of stale Sender IDs for the group is equal to N, then the Group Manager deletes the oldest set.

* The Group Manager rekeys the group. This includes also distributing the set of stale Sender IDs associated with the version V of the group keying material (see Section 7).

* After completing the group rekeying, the Group Manager creates an empty set of stale Sender IDs, as associated with the version V' of the group keying material.

8. Interface at the Group Manager

The Group Manager provides the interface defined in Section 4.1 of [I-D.ietf-ace-key-groupcomm], with the additional sub-resources defined from Section 8.1 to Section 8.3 of this document.

Furthermore, Section 8.4 provides a summary of the CoAP methods admitted to access different resources at the Group Manager, for nodes with different roles in the group or as non members (REQ11).

The GROUPNAME segment of the URI path MUST match with the group name specified in the scope entry of the Access Token scope (i.e., 'gname' in Section 3.1 of [I-D.ietf-ace-key-groupcomm]) (REQ7).

The Resource Type (rt=) Link Target Attribute value "core.osc.gm" is registered in Section 16.11 (REQ10), and can be used to describe group-membership resources and its sub-resources at a Group Manager, e.g., by using a link-format document [RFC6690].

Applications can use this common resource type to discover links to group-membership resources for joining OSCORE groups, e.g., by using the approach described in [I-D.tiloca-core-oscore-discovery].

8.1. ace-group/GROUPNAME/active

This resource implements a GET handler.

8.1.1. GET Handler

The handler expects a GET request.

In addition to what is defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm], the handler verifies that the requesting Client is a current member of the group. If the verification fails, the KDC MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler replies with a 2.05 (Content) response, specifying the current status of the group, i.e., active or inactive. The payload of the response is formatted as defined in Section 9.9.

The method to set the current group status is out of the scope of this document, and is defined for the administrator interface of the Group Manager specified in [I-D.ietf-ace-oscore-gm-admin].

8.2. ace-group/GROUPNAME/verif-data

This resource implements a GET handler.

8.2.1. GET Handler

The handler expects a GET request.

In addition to what is defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm], the Group Manager performs the following checks.

If the requesting Client is a current group member, the Group Manager MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 8 ("Operation permitted only to signature verifiers").

If GROUPNAME denotes a pairwise-only group, the Group Manager MUST reply with a 4.00 (Bad Request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 7 ("Signatures not used in the group").

If all verifications succeed, the handler replies with a 2.05 (Content) response, specifying data that allow also an external signature verifier to verify signatures of messages protected with the group mode and sent to the group (see Sections 3.1 and 8.5 of [I-D.ietf-core-oscore-groupcomm]). The response MUST have Content-Format set to application/ace-groupcomm+cbor. The payload of the response is a CBOR map, which is formatted as defined in Section 9.6.

8.3. ace-group/GROUPNAME/stale-sids

This resource implements a FETCH handler.

8.3.1. FETCH Handler

The handler expects a FETCH request, whose payload specifies a version number of the group keying material, encoded as an unsigned CBOR integer.

In addition to what is defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm], the handler verifies that the requesting Client is a current member of the group. If the verification fails, the Group Manager MUST reply with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 0 ("Operation permitted only to group members").

If all verifications succeed, the handler replies with a 2.05 (Content) response, specifying data that allow the requesting Client to delete the Recipient Contexts and authentication credentials associated with former members of the group (see Section 3.2 of [I-D.ietf-core-oscore-groupcomm]. The payload of the response is formatted as defined in Section 11.3.1.

8.4. Admitted Methods

The table in Figure 2 summarizes the CoAP methods admitted to access different resources at the Group Manager, for (non-)members of a group with group name GROUPNAME, and considering different roles. The last two rows of the table apply to a node with node name NODENAME.

Resource	Type1	Type2	Type3	Type4
ace-group/	F	F	F	F
ace-group/GROUPNAME/	G Po	G Po	Po *	Po
ace-group/GROUPNAME/active	G	G	-	-
ace-group/GROUPNAME/verif-data	-	-	G	-
ace-group/GROUPNAME/creds	G F	G F	G F	-
ace-group/GROUPNAME/kdc-cred	G	G	G	-
ace-group/GROUPNAME/stale-sids	F	F	-	-
ace-group/GROUPNAME/policies	G	G	-	-
ace-group/GROUPNAME/num	G	G	-	-
ace-group/GROUPNAME/nodes/ NODENAME	G Pu D	G D	-	-
ace-group/GROUPNAME/nodes/ NODENAME/pub-key	Po	-	-	-

CoAP methods: G = GET; F = FETCH; Po = POST; Pu = PUT; D = DELETE

Type1 = Member as Requester and/or Responder
Type2 = Member as Monitor
Type3 = Non-member (authorized to be signature verifier)
(*) = cannot join the group as signature verifier
Type4 = Non-member (not authorized to be signature verifier)

Figure 2: Admitted CoAP Methods on the Group Manager Resources

8.4.1. Signature Verifiers

Just like any candidate group member, a signature verifier provides the Group Manager with an Access Token, as described in Section 5.3. However, unlike candidate group members, it does not join any OSCORE group, i.e., it does not perform the joining process defined in Section 6.

After successfully transferring an Access Token to the Group Manager, a signature verifier is allowed to perform only some operations as non-member of a group, and only for the OSCORE groups specified in the validated Access Token. These are the operations specified in Section 9.3, Section 9.5, Section 9.6 and Section 9.10.

Consistently, in case a node is not a member of the group with group name GROUPNAME and is authorized to be only signature verifier for that group, the Group Manager MUST reply with a 4.03 (Forbidden) error response if that node attempts to access any other endpoint than: /ace-group; ace-group/GROUPNAME/verif-data; /ace-group/GROUPNAME/creds; and ace-group/GROUPNAME/kdc-cred.

8.5. Operations Supported by Clients

Building on what is defined in Section 4.1.1 of [I-D.ietf-ace-key-groupcomm], and with reference to the resources at the Group Manager newly defined earlier in Section 8 of this document, it is expected that a Client minimally supports also the following set of operations and corresponding interactions with the Group Manager (REQ12).

* GET request to ace-group/GROUPNAME/active, in order to check the current status of the group.

* GET request to ace-group/GROUPNAME/verif-data, in order for a signature verifier to retrieve data required to verify signatures of messages protected with the group mode of Group OSCORE and sent to a group (see Sections 3.1 and 8.5 of [I-D.ietf-core-oscore-groupcomm]). Note that this operation is relevant to support only to signature verifiers.

* FETCH request to ace-group/GROUPNAME/stale-sids, in order to retrieve from the Group Manager the data required to delete some of the stored group members' authentication credentials and associated Recipient Contexts (see Section 8.3.1). This data is provided as an aggregated set of stale Sender IDs, which are used as specified in Section 11.3.

9. Additional Interactions with the Group Manager

This section defines the possible interactions with the Group Manager, in addition to the group joining specified in Section 6.

9.1. Retrieve Updated Keying Material

At some point, a group member considers the Group OSCORE Security Context invalid and to be renewed. This happens, for instance, after a number of unsuccessful security processing of incoming messages from other group members, or when the Security Context expires as specified by the 'exp' parameter of the Join Response.

When this happens, the group member retrieves updated security parameters and group keying material. This can occur in the two different ways described below.

9.1.1. Get Group Keying Material

If the group member wants to retrieve only the latest group keying material, it sends a Key Distribution Request to the Group Manager.

That is, it sends a CoAP GET request to the endpoint /ace-group/GROUPNAME at the Group Manager.

The Group Manager processes the Key Distribution Request according to Section 4.3.2 of [I-D.ietf-ace-key-groupcomm]. The Key Distribution Response is formatted as defined in Section 4.3.2 of [I-D.ietf-ace-key-groupcomm], with the following additions.

* The 'key' parameter is formatted as defined in Section 6.3 of this document, with the difference that it does not include the 'group_SenderId' parameter.

* The 'exp' parameter MUST be present.

* The 'ace_groupcomm_profile' parameter MUST be present and has value coap_group_oscore_app.

Upon receiving the Key Distribution Response, the group member retrieves the updated security parameters and group keying material, and, if they differ from the current ones, uses them to set up the new Group OSCORE Security Context as described in Section 2 of [I-D.ietf-core-oscore-groupcomm].

9.1.2. Get Group Keying Material and OSCORE Sender ID

If the group member wants to retrieve the latest group keying material as well as the OSCORE Sender ID that it has in the OSCORE group, it sends a Key Distribution Request to the Group Manager.

That is, it sends a CoAP GET request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME at the Group Manager.

The Group Manager processes the Key Distribution Request according to Section 4.8.1 of [I-D.ietf-ace-key-groupcomm]. The Key Distribution Response is formatted as defined in Section 4.8.1 of [I-D.ietf-ace-key-groupcomm], with the following additions.

* The 'key' parameter is formatted as defined in Section 6.3 of this document. If the requesting group member has exclusively the role of monitor, then the 'key' parameter does not include the 'group_SenderId'.

  Note that, in any other case, the current Sender ID of the group member is not specified as a separate parameter, but rather specified by 'group_SenderId' within the 'key' parameter.

* The 'exp' parameter MUST be present.

Upon receiving the Key Distribution Response, the group member retrieves the updated security parameters, group keying material and Sender ID, and, if they differ from the current ones, uses them to set up the new Group OSCORE Security Context as described in Section 2 of [I-D.ietf-core-oscore-groupcomm].

9.2. Request to Change Individual Keying Material

As discussed in Section 2.5.2 of [I-D.ietf-core-oscore-groupcomm], a group member may at some point exhaust its Sender Sequence Numbers in the OSCORE group.

When this happens, the group member MUST send a Key Renewal Request message to the Group Manager, as per Section 4.8.2.1 of [I-D.ietf-ace-key-groupcomm]. That is, it sends a CoAP PUT request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME at the Group Manager.

Upon receiving the Key Renewal Request, the Group Manager processes it as defined in Section 4.8.2 of [I-D.ietf-ace-key-groupcomm], with the following additions.

The Group Manager MUST return a 5.03 (Service Unavailable) response in case the OSCORE group identified by GROUPNAME is currently inactive (see Section 8.1). The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 9 ("Group currently not active").

Otherwise, the Group Manager performs one of the following actions.

1. If the requesting group member has exclusively the role of monitor, the Group Manager replies with a 4.03 (Forbidden) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 1 ("Request inconsistent with the current roles").

2. Otherwise, the Group Manager takes one of the following actions.

   * The Group Manager rekeys the OSCORE group. That is, the Group Manager generates new group keying material for that group (see Section 11), and replies to the group member with a group rekeying message as defined in Section 11, providing the new group keying material. Then, the Group Manager rekeys the rest of the OSCORE group, as discussed in Section 11.

     The Group Manager SHOULD perform a group rekeying only if already scheduled to occur shortly, e.g., according to an application-specific rekeying period or scheduling, or as a reaction to a recent change in the group membership. In any other case, the Group Manager SHOULD NOT rekey the OSCORE group when receiving a Key Renewal Request (OPT12).

   * The Group Manager determines and assigns a new OSCORE Sender ID for that group member, and replies with a Key Renewal Response formatted as defined in Section 4.8.2 of [I-D.ietf-ace-key-groupcomm]. The CBOR Map in the response payload includes a single parameter 'group_SenderId' defined in Section 16.3 of this document, specifying the new Sender ID of the group member encoded as a CBOR byte string.

     Consistently with Section 2.5.3.1 of [I-D.ietf-core-oscore-groupcomm], the Group Manager MUST assign a new Sender ID that has not been used in the OSCORE group since the latest time when the current Gid value was assigned to the group.

Furthermore, the Group Manager MUST add the old, relinquished Sender ID of the group member to the most recent set of stale Sender IDs for the group (see Section 7.1).

The Group Manager MUST return a 5.03 (Service Unavailable) response in case there are currently no Sender IDs available to assign in the OSCORE group. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 4 ("No available node identifiers").

9.3. Retrieve Authentication Credentials of Group Members

A group member or a signature verifier may need to retrieve the authentication credentials of (other) group members. To this end, the group member or signature verifier sends an Authentication Credential Request message to the Group Manager, as per Sections 4.4.1.1 and 4.4.2.1 of [I-D.ietf-ace-key-groupcomm]. That is, it sends the request to the endpoint /ace-group/GROUPNAME/creds at the Group Manager.

If the Authentication Credential Request uses the method FETCH, the Authentication Credential Request is formatted as defined in Section 4.4.1 of [I-D.ietf-ace-key-groupcomm]. That is:

* Each element (if any) of the inner CBOR array 'role_filter' is formatted as in the inner CBOR array 'role_filter' of the 'get_creds' parameter of the Join Request when the parameter value is not the CBOR simple value "null" (0xf6) (see Section 6.1).

* Each element (if any) of the inner CBOR array 'id_filter' is a CBOR byte string, which encodes the OSCORE Sender ID of the group member for which the associated authentication credential is requested (REQ25).

Upon receiving the Authentication Credential Request, the Group Manager processes it as per Section 4.4.1 or Section 4.4.2 of [I-D.ietf-ace-key-groupcomm], depending on the request method being FETCH or GET, respectively. Additionally, if the Authentication Credential Request uses the method FETCH, the Group Manager silently ignores node identifiers included in the 'get_creds' parameter of the request that are not associated with any current group member (REQ26).

The success Authentication Credential Response is formatted as defined in Section 4.4.1 or Section 4.4.2 of [I-D.ietf-ace-key-groupcomm], depending on the request method being FETCH or GET, respectively.

9.4. Upload a New Authentication Credential

A group member may need to provide the Group Manager with its new authentication credential to use in the group from then on, hence replacing the current one. This can be the case, for instance, if the signature or ECDH algorithm and possible associated parameters used in the OSCORE group have been changed, and the current authentication credential is not compatible with them.

To this end, the group member sends an Authentication Credential Update Request message to the Group Manager, as per Section 4.9.1.1 of [I-D.ietf-ace-key-groupcomm], with the following addition.

* The group member computes the proof-of-possession (PoP) evidence included in 'client_cred_verify' in the same way taken when preparing a Join Request for the OSCORE group in question, as defined in Section 6.1 (REQ14).

That is, the group member sends a CoAP POST request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME/cred at the Group Manager.

Upon receiving the Authentication Credential Update Request, the Group Manager processes it as per Section 4.9.1 of [I-D.ietf-ace-key-groupcomm], with the following additions.

* The N_S challenge used to build the proof-of-possession input is computed as defined in Section 6.1.1 (REQ15).

* The Group Manager verifies the PoP challenge included in 'client_cred_verify' in the same way as when processing a Join Request for the OSCORE group in question, as defined in Section 6.2 (REQ14).

* The Group Manager MUST return a 5.03 (Service Unavailable) response in case the OSCORE group identified by GROUPNAME is currently inactive (see Section 8.1). The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 9 ("Group currently not active").

* If the requesting group member has exclusively the role of monitor, the Group Manager replies with a 4.00 (Bad request) error response. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 1 ("Request inconsistent with the current roles").

* If the request is successfully processed, the Group Manager stores the association between i) the new authentication credential of the group member; and ii) the Group Identifier (Gid), i.e., the OSCORE ID Context, associated with the OSCORE group together with the OSCORE Sender ID assigned to the group member in the group. The Group Manager MUST keep this association updated over time.

9.5. Retrieve the Group Manager's Authentication Credential

A group member or a signature verifier may need to retrieve the authentication credential of the Group Manager. To this end, the requesting Client sends a KDC Authentication Credential Request message to the Group Manager.

That is, it sends a CoAP GET request to the endpoint /ace-group/GROUPNAME/kdc-cred at the Group Manager defined in Section 4.5.1.1 of [I-D.ietf-ace-key-groupcomm], where GROUPNAME is the name of the OSCORE group.

In addition to what is defined in Section 4.5.1 of [I-D.ietf-ace-key-groupcomm], the Group Manager MUST respond with a 4.00 (Bad Request) error response, if the requesting Client is not a current group member and GROUPNAME denotes a pairwise-only group. The response MUST have Content-Format set to application/ace-groupcomm+cbor and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. The value of the 'error' field MUST be set to 7 ("Signatures not used in the group").

The payload of the 2.05 (Content) KDC Authentication Credential Response is a CBOR map, which is formatted as defined in Section 4.5.1 of [I-D.ietf-ace-key-groupcomm]. The Group Manager specifies the parameters 'kdc_cred', 'kdc_nonce' and 'kdc_challenge' as defined for the Join Response in Section 6.3 of this document. This especially applies to the computing of the proof-of-possession (PoP) evidence included in 'kdc_cred_verify' (REQ21).

Upon receiving a 2.05 (Content) KDC Authentication Credential Response, the requesting Client retrieves the Group Manager's authentication credential from the 'kdc_cred' parameter, and proceeds as defined in Section 4.5.1.1 of [I-D.ietf-ace-key-groupcomm]. The

requesting Client verifies the PoP evidence included in 'kdc_cred_verify' by means of the same method used when processing the Join Response, as defined in Section 6.3 of this document (REQ21).

Note that a signature verifier would not receive a successful response from the Group Manager, in case GROUPNAME denotes a pairwise-only group.

9.6. Retrieve Signature Verification Data

A signature verifier may need to retrieve data required to verify signatures of messages protected with the group mode and sent to a group (see Sections 3.1 and 8.5 of [I-D.ietf-core-oscore-groupcomm]). To this end, the signature verifier sends a Signature Verification Data Request message to the Group Manager.

That is, it sends a CoAP GET request to the endpoint /ace-group/GROUPNAME/verif-data at the Group Manager defined in Section 8.2 of this document, where GROUPNAME is the name of the OSCORE group.

The payload of the 2.05 (Content) Signature Verification Data Response is a CBOR map, which has the format used for the Join Response message in Section 6.3, with the following differences.

* From the Join Response message, only the parameters 'gkty', 'key', 'num', 'exp' and 'ace_groupcomm_profile' are present. The 'key' parameter includes only the following data.

  - The parameters 'hkdf', 'contextId', 'cred_fmt', 'sign_enc_alg', 'sign_alg', 'sign_params'. These parameters MUST be present.

  - The parameters 'alg' and 'ecdh_alg'. These parameters MUST NOT be present if the group is a signature-only group. Otherwise, they MUST be present.

* The parameter 'group_enc_key' is also included, with CBOR label defined in Section 16.3. This parameter specifies the Group Encryption Key of the OSCORE Group, encoded as a CBOR byte string. The Group Manager derives the Group Encryption Key from the group keying material, as per Section 2.1.6 of [I-D.ietf-core-oscore-groupcomm]. This parameter MUST be present.

In order to verify signatures in the group (see Section 8.5 of [I-D.ietf-core-oscore-groupcomm]), the signature verifier relies on: the data retrieved from the 2.05 (Content) Signature Verification Data Response; the public keys of the group members signing the

messages to verify, retrieved from those members' authentication credentials that can be obtained as defined in Section 9.3; and the public key of the Group Manager, retrieved from the Group Manager's authentication credential that can be obtained as defined in Section 9.5.

Figure 3 gives an overview of the exchange described above, while Figure 4 shows an example of Signature Verification Data Request-Response.

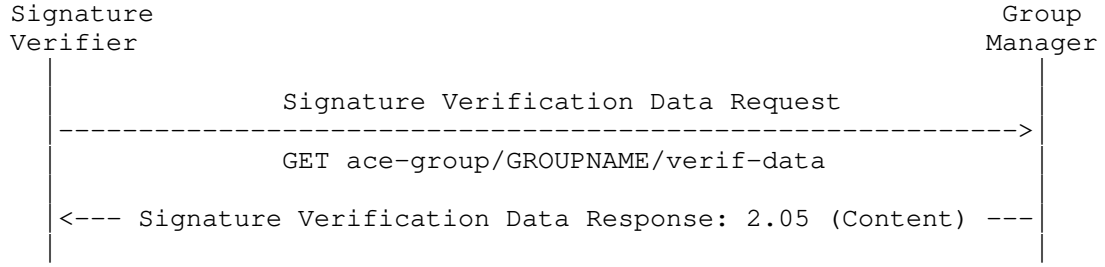


Figure 3: Message Flow of Signature Verification Data Request-Response

```
Request:

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "verif-data"
Payload: -

Response:

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation, with GROUPCOMM_KEY_TBD
         and PROFILE_TBD being CBOR integers, while GROUP_ENC_KEY
         being a CBOR byte string):
 {
   "gkty": GROUPCOMM_KEY_TBD,
   "key": {
     "hkdf": 5,                      ; HMAC 256/256
     "contextId": h'37fc',
     "cred_fmt": 33,                 ; x5chain
     "sign_enc_alg": 10,             ; AES-CCM-16-64-128
     "sign_alg": -8,                 ; EdDSA
     "sign_params": [[1], [1, 6]]    ; [[OKP], [OKP, Ed25519]]
   },
   "num": 12,
   "exp": 1609459200,
   "ace_groupcomm_profile": PROFILE_TBD,
   "group_enc_key": GROUP_ENC_KEY
 }
```

Figure 4: Example of Signature Verification Data Request-Response

9.7. Retrieve the Group Policies

A group member may request the current policies used in the OSCORE group. To this end, the group member sends a Policies Request, as per Section 4.6.1.1 of [I-D.ietf-ace-key-groupcomm]. That is, it sends a CoAP GET request to the endpoint /ace-group/GROUPNAME/policies at the Group Manager, where GROUPNAME is the name of the OSCORE group.

Upon receiving the Policies Request, the Group Manager processes it as per Section 4.6.1 of [I-D.ietf-ace-key-groupcomm]. The success Policies Response is formatted as defined in Section 4.6.1 of [I-D.ietf-ace-key-groupcomm].

9.8. Retrieve the Keying Material Version

A group member may request the current version of the keying material used in the OSCORE group. To this end, the group member sends a Version Request, as per Section 4.7.1.1 of [I-D.ietf-ace-key-groupcomm]. That is, it sends a CoAP GET request to the endpoint /ace-group/GROUPNAME/num at the Group Manager, where GROUPNAME is the name of the OSCORE group.

Upon receiving the Version Request, the Group Manager processes it as per Section 4.7.1 of [I-D.ietf-ace-key-groupcomm]. The success Version Response is formatted as defined in Section 4.7.1 of [I-D.ietf-ace-key-groupcomm].

9.9. Retrieve the Group Status

A group member may request the current status of the OSCORE group, i.e., active or inactive. To this end, the group member sends a Group Status Request to the Group Manager.

That is, the group member sends a CoAP GET request to the endpoint /ace-group/GROUPNAME/active at the Group Manager defined in Section 8.1 of this document, where GROUPNAME is the name of the OSCORE group.

The payload of the 2.05 (Content) Group Status Response includes the CBOR simple value "true" (0xf5) if the group is currently active, or the CBOR simple value "false" (0xf4) otherwise. The group is considered active if it is set to allow new members to join, and if communication within the group is fine to happen.

Upon learning from a 2.05 (Content) response that the group is currently inactive, the group member SHOULD stop taking part in communications within the group, until it becomes active again.

Upon learning from a 2.05 (Content) response that the group has become active again, the group member can resume taking part in communications within the group.

Figure 5 gives an overview of the exchange described above, while Figure 6 shows an example of Group Status Request-Response.

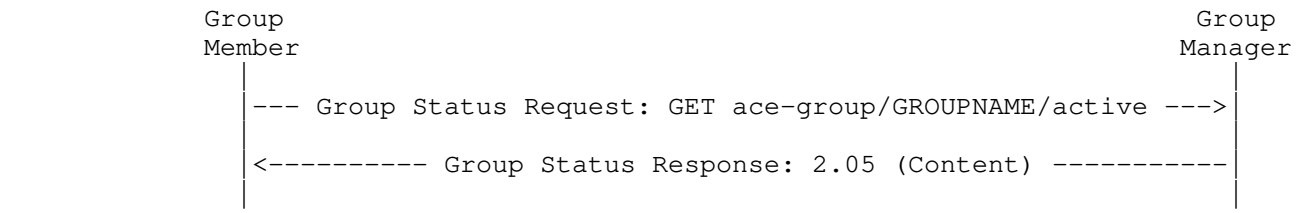


Figure 5: Message Flow of Group Status Request-Response

Request:

Header: GET (Code=0.01)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "active"
Payload: -

Response:

Header: Content (Code=2.05)
Payload (in CBOR diagnostic notation):
  true

Figure 6: Example of Group Status Request-Response

9.10. Retrieve Group Names

A node may want to retrieve from the Group Manager the group name and the URI of the group-membership resource of a group. This is relevant in the following cases.

* Before joining a group, a joining node may know only the current Group Identifier (Gid) of that group, but not the group name and the URI to the group-membership resource.

* As current group member in several groups, the node has missed a previous group rekeying in one of them (see Section 11). Hence, it retains stale keying material and fails to decrypt received messages exchanged in that group.

Such messages do not provide a direct hint to the correct group name, that the node would need in order to retrieve the latest keying material and authentication credentials from the Group Manager (see Section 9.1.1, Section 9.1.2 and Section 9.3). However, such messages may specify the current Gid of the group, as value of the 'kid_context' field of the OSCORE CoAP option (see Section 6.1 of [RFC8613] and Section 4.2 of [I-D.ietf-core-oscore-groupcomm]).

* As signature verifier, the node also refers to a group name for retrieving the required authentication credentials from the Group Manager (see Section 9.3). As discussed above, intercepted messages do not provide a direct hint to the correct group name, while they may specify the current Gid of the group, as value of the 'kid_context' field of the OSCORE CoAP option. In such a case, upon intercepting a message in the group, the node requires to correctly map the Gid currently used in the group with the invariant group name.

  Furthermore, since it is not a group member, the node does not take part to a possible group rekeying. Thus, following a group rekeying and the consequent change of Gid in a group, the node would retain the old Gid value and cannot correctly associate intercepted messages to the right group, especially if acting as signature verifier in several groups. This in turn prevents the efficient verification of signatures, and especially the retrieval of required, new authentication credentials from the Group Manager.

In either case, the node only knows the current Gid of the group, as learned from received or intercepted messages exchanged in the group. As detailed below, the node can contact the Group Manager, and request the group name and URI to the group-membership resource corresponding to that Gid. Then, it can use that information to join the group, or get the latest keying material as a current group member, or retrieve authentication credentials used in the group as a signature verifier. To this end, the node sends a Group Name and URI Retrieval Request, as per Section 4.2.1.1 of [I-D.ietf-ace-key-groupcomm].

That is, the node sends a CoAP FETCH request to the endpoint /ace-group at the Group Manager formatted as defined in Section 4.2.1 of [I-D.ietf-ace-key-groupcomm]. Each element of the CBOR array 'gid' is a CBOR byte string (REQ13), which encodes the Gid of the group for which the group name and the URI to the group-membership resource are requested.

Upon receiving the Group Name and URI Retrieval Request, the Group Manager processes it as per Section 4.2.1 of [I-D.ietf-ace-key-groupcomm]. The success Group Name and URI Retrieval Response is formatted as defined in Section 4.2.1 of [I-D.ietf-ace-key-groupcomm]. Each element of the CBOR array 'gid' is a CBOR byte string (REQ13), which encodes the Gid of the group for which the group name and the URI to the group-membership resource are provided.

For each of its groups, the Group Manager maintains an association between the group name and the URI to the group-membership resource on one hand, and only the current Gid for that group on the other hand. That is, the Group Manager does not maintain an association between the former pair and any other Gid for that group than the current, most recent one.

Figure 7 gives an overview of the exchanges described above, while Figure 8 shows an example of Group Name and URI Retrieval Request-Response.

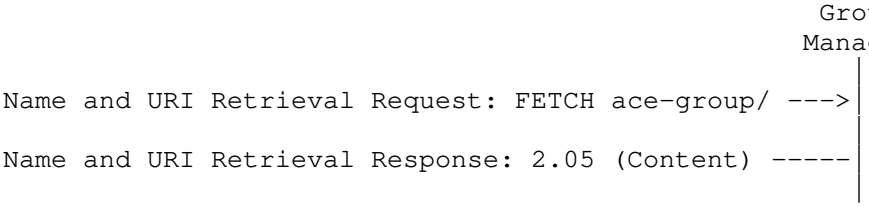


Figure 7: Message Flow of Group Name and URI Retrieval Request-Response

```
Request:

Header: FETCH (Code=0.05)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
  {
    "gid": [h'37fc', h'84bd']
  }

Response:

Header: Content (Code=2.05)
Content-Format: "application/ace-groupcomm+cbor"
Payload (in CBOR diagnostic notation):
  {
    "gid": [h'37fc', h'84bd'],
    "gname": ["g1", "g2"],
    "guri": ["ace-group/g1", "ace-group/g2"]
  }
```

Figure 8: Example of Group Name and URI Retrieval Request-Response

9.11. Leave the Group

A group member may request to leave the OSCORE group. To this end, the group member sends a Group Leaving Request, as per Section 4.8.3.1 of [I-D.ietf-ace-key-groupcomm]. That is, it sends a CoAP DELETE request to the endpoint /ace-group/GROUPNAME/nodes/NODENAME at the Group Manager.

Upon receiving the Group Leaving Request, the Group Manager processes it as per Section 4.8.3 of [I-D.ietf-ace-key-groupcomm]. Then, the Group Manager performs the follow-up actions defined in Section 10 of this document.

10. Removal of a Group Member

Other than after a spontaneous request to the Group Manager as described in Section 9.11, a node may be forcibly removed from the OSCORE group, e.g., due to expired or revoked authorization.

In either case, if the Group Manager reassigns Gid values during the group's lifetime (see Section 3.2.1.1 of [I-D.ietf-core-oscore-groupcomm]), the Group Manager "forgets" the Birth Gid currently associated with the leaving node in the OSCORE group. This was stored following the Join Response sent to that node, after its latest (re-)joining of the OSCORE group (see Section 6.3).

If any of the two conditions below holds, the Group Manager MUST inform the leaving node of its eviction as follows. If both conditions hold, the Group Manager MUST inform the leaving node by using only the method corresponding to one of either conditions.

* If, upon joining the group (see Section 6.1), the leaving node specified a URI in the 'control_uri' parameter defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], the Group Manager sends a DELETE request targeting the URI specified in the 'control_uri' parameter (OPT7).

* If the leaving node has been observing the associated resource at ace-group/GROUPNAME/nodes/NODENAME, the Group Manager sends an unsolicited 4.04 (Not Found) error response to the leaving node, as specified in Section 4.3.2 of [I-D.ietf-ace-key-groupcomm].

Furthermore, the Group Manager might intend to evict all the current group members from the group at once. In such a case, if the Join Responses sent by the Group Manager to nodes joining the group (see Section 6.3) specify a URI in the 'control_group_uri' parameter defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], then the Group Manager MUST additionally send a DELETE request targeting the URI specified in the 'control_group_uri' parameter (OPT10).

If the leaving node has not exclusively the role of monitor, the Group Manager performs the following actions.

* The Group Manager frees the OSCORE Sender ID value of the leaving node. This value MUST NOT become available for possible upcoming joining nodes in the same group, until the group has been rekeyed and assigned a new Group Identifier (Gid).

* The Group Manager MUST add the relinquished Sender ID of the leaving node to the most recent set of stale Sender IDs for the group (see Section 7.1).

* The Group Manager cancels the association between, on one hand, the authentication credential of the leaving node and, on the other hand, the Gid associated with the OSCORE group together with the freed Sender ID value. The Group Manager deletes the

authentication credential of the leaving node, if that authentication credential has no remaining association with any pair (Gid, Sender ID).

Then, the Group Manager MUST generate updated security parameters and group keying material, and provide it to the remaining group members (see Section 11). As a consequence, the leaving node is not able to acquire the new security parameters and group keying material distributed after its leaving.

The same considerations from Section 5 of [I-D.ietf-ace-key-groupcomm] apply here as well, considering the Group Manager acting as KDC.

11. Group Rekeying Process

In order to rekey the OSCORE group, the Group Manager distributes a new Group Identifier (Gid), i.e., a new OSCORE ID Context; a new OSCORE Master Secret; and, optionally, a new OSCORE Master Salt for that group. When doing so, the Group Manager MUST increment the version number of the group keying material, before starting its distribution.

As per Section 3.2.1.1 of [I-D.ietf-core-oscore-groupcomm], the Group Manager MAY reassign a Gid to the same group over that group's lifetime, e.g., once the whole space of Gid values has been used for the group in question. If the Group Manager supports reassignment of Gid values and performs it in a group, then the Group Manager additionally takes the following actions.

* Before rekeying the group, the Group Manager MUST check if the new Gid to be distributed coincides with the Birth Gid of any of the current group members (see Section 6.3).

* If any of such "elder members" is found in the group, the Group Manager MUST evict them from the group. That is, the Group Manager MUST terminate their membership and MUST rekey the group in such a way that the new keying material is not provided to those evicted elder members. This also includes adding their relinquished Sender IDs to the most recent set of stale Sender IDs for the group (see Section 7.1), before rekeying the group.

Until a further following group rekeying, the Group Manager MUST store the list of those latest-evicted elder members. If any of those nodes re-joins the group before a further following group rekeying occurs, the Group Manager MUST NOT rekey the group upon their re-joining. When one of those nodes re-joins the group, the Group Manager can rely, e.g., on the ongoing secure communication association to recognize the node as included in the stored list.

Across the rekeying execution, the Group Manager MUST preserve the same unchanged OSCORE Sender IDs for all group members intended to remain in the group. This avoids affecting the retrieval of authentication credentials from the Group Manager and the verification of group messages.

The Group Manager MUST support the "Point-to-Point" group rekeying scheme registered in Section 11.12 of [I-D.ietf-ace-key-groupcomm], as per the detailed use defined in Section 11.1 of this document. Future specifications may define how this application profile can use alternative group rekeying schemes, which MUST comply with the functional steps defined in Section 3.2 of [I-D.ietf-core-oscore-groupcomm]. The Group Manager MUST indicate the use of such an alternative group rekeying scheme to joining nodes, by means of the 'group_rekeying' parameter included in Join Response messages (see Section 6.3).

It is RECOMMENDED that the Group Manager gets confirmation of successful distribution from the group members, and admits a maximum number of individual retransmissions to non-confirming group members. Once completed the group rekeying process, the Group Manager creates a new empty set of stale Sender IDs associated with the version of the newly distributed group keying material (see Section 7.1).

In case the rekeying terminates and some group members have not received the new keying material, they will not be able to correctly process following secured messages exchanged in the group. These group members will eventually contact the Group Manager, in order to retrieve the current keying material and its version.

Some of these group members may be in multiple groups, each associated with a different Group Manager. When failing to correctly process messages secured with the new keying material, these group members may not have sufficient information to determine which exact Group Manager they should contact, in order to retrieve the current keying material they are missing.

If the Gid is formatted as described in Appendix C of [I-D.ietf-core-oscore-groupcomm], the Group Prefix can be used as a hint to determine the right Group Manager, as long as no collisions

among Group Prefixes are experienced. Otherwise, a group member needs to contact the Group Manager of each group, e.g., by first requesting only the version of the current group keying material (see Section 9.8) and then possibly requesting the current keying material (see Section 9.1.1).

Furthermore, some of these group members can be in multiple groups, all of which are associated with the same Group Manager. In this case, these group members may also not have sufficient information to determine which exact group they should refer to, when contacting the right Group Manager. Hence, they need to contact a Group Manager multiple times, i.e., separately for each group they belong to and associated with that Group Manager.

Section 11.2 defines the actions performed by a group member upon receiving the new group keying material. Section 11.3 discusses how a group member can realize that it has missed one or more rekeying instances, and the actions it is accordingly required to take.

11.1. Sending Rekeying Messages

When using the "Point-to-Point" group rekeying scheme, the group rekeying messages MUST have Content-Format set to application/ace-groupcomm+cbor and have the same format used for the Join Response message in Section 6.3, with the following differences. Note that this extends the minimal content of a rekeying message as defined in Section 6 of [I-D.ietf-ace-key-groupcomm] (OPT14).

* From the Join Response, only the parameters 'gkty', 'key', 'num', 'exp', and 'ace_groupcomm_profile' are present. The 'key' parameter includes only the following data.
  - The 'ms' parameter, specifying the new OSCORE Master Secret value. This parameter MUST be present.
  - The 'contextId' parameter, specifying the new Gid to use as OSCORE ID Context value. This parameter MUST be present.
  - The 'salt' value, specifying the new OSCORE Master Salt value. This parameter MAY be present.
* The parameter 'stale_node_ids' MUST also be included, with CBOR label defined in Section 16.3. This parameter is encoded as a CBOR array, where each element is encoded as a CBOR byte string. The order of elements in the CBOR array is irrelevant. The parameter is populated as follows.
  - The Group Manager creates an empty CBOR array ARRAY.

- The Group Manager considers the most recent set of stale Sender IDs for the group (see Section 7.1), i.e., the set X associated with the current version of the group keying material about to be relinquished.

- For each Sender ID in X, the Group Manager encodes it as a CBOR byte string and adds the result to ARRAY.

- The parameter 'stale_node_ids' takes ARRAY as value.

* The parameters 'creds', 'peer_roles' and 'peer_identifiers' SHOULD be present, if the group rekeying is performed due to one or multiple Clients that have requested to join the group. Following the same semantics used in the Join Response message (see Section 6.3), the three parameters specify the authentication credential, roles in the group and node identifier of each of the Clients that have requested to join the group. The Group Manager MUST NOT include a non-empty subset of these three parameters.

The Group Manager separately sends a group rekeying message formatted as defined above to each group member to be rekeyed.

Each rekeying message MUST be secured with the pairwise secure communication association between the Group Manager and the group member used during the joining process. Each rekeying message can target the 'control_uri' URI path defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm] (OPT7), if provided by the intended recipient upon joining the group (see Section 6.1).

This distribution approach requires group members to act (also) as servers, in order to correctly handle unsolicited group rekeying messages from the Group Manager. If a group member and the Group Manager use OSCORE [RFC8613] to secure their pairwise communications, the group member MUST create a Replay Window in its own Recipient Context upon establishing the OSCORE Security Context with the Group Manager, e.g., by means of the OSCORE profile of ACE [RFC9203].

Group members and the Group Manager SHOULD additionally support alternative distribution approaches that do not require group members to act (also) as servers. A number of such approaches are defined in Section 6 of [I-D.ietf-ace-key-groupcomm]. In particular, a group member may use CoAP Observe [RFC7641] and subscribe for updates to the group-membership resource of the group, at the endpoint /ace-group/GROUPNAME/ of the Group Manager (see Section 6.1 of [I-D.ietf-ace-key-groupcomm]). Alternatively, a full-fledged Pub-Sub model can be considered [I-D.ietf-core-coap-pubsub], where the Group Manager publishes to a rekeying topic hosted at a Broker, while the group members subscribe to such topic (see Section 6.2 of [I-D.ietf-ace-key-groupcomm]).

11.2. Receiving Rekeying Messages

After having received the new group keying material, a group member proceeds as follows. Unless otherwise specified, the following is independent of the specifically used group rekeying scheme.

The group member considers the stale Sender IDs received from the Group Manager. If the "Point-to-Point" group rekeying scheme as detailed in Section 11.1 is used, the stale Sender IDs are specified by the 'stale_node_ids' parameter.

After that, as per Section 3.2 of [I-D.ietf-core-oscore-groupcomm], the group member MUST remove every authentication credential associated with a stale Sender ID from its list of group members' authentication credentials used in the group, and MUST delete each of its Recipient Contexts used in the group whose corresponding Recipient ID is a stale Sender ID.

Then, the following cases can occur, based on the version number V' of the new group keying material distributed through the rekeying process. If the "Point-to-Point" group rekeying scheme as detailed in Section 11.1 is used, this information is specified by the 'num' parameter.

* The group member has not missed any group rekeying. That is, the old keying material stored by the group member has version number V, while the received new keying material has version number V' = (V + 1). In such a case, the group member simply installs the new keying material and derives the corresponding new Security Context.

* The group member has missed one or more group rekeying instances. That is, the old keying material stored by the group member has version number V, while the received new keying material has version number V' > (V + 1). In such a case, the group member MUST proceed as defined in Section 11.3.

* The group member has received keying material not newer than the stored one. That is, the old keying material stored by the group member has version number V, while the received keying material has version number V' < (V + 1). In such a case, the group member MUST ignore the received rekeying messages and MUST NOT install the received keying material.

11.3. Missed Rekeying Instances

A group member can realize to have missed one or more rekeying instances in one of the ways discussed below. In the following, V denotes the version number of the old keying material stored by the group member, while V' denotes the version number of the latest, possibly just distributed, keying material.

a. The group member has participated to a rekeying process that has distributed new keying material with version number V' > (V + 1), as discussed in Section 11.2.

b. The group member has obtained the latest keying material from the Group Manager, as a response to a Key Distribution Request (see Section 9.1.1) or to a Join Request when re-joining the group (see Section 6.1). That is, V is different than V' specified by the 'num' parameter in the response.

c. The group member has obtained the authentication credentials of other group members, through an Authentication Credential Request-Response exchange with the Group Manager (see Section 9.3). That is, V is different than V' specified by the 'num' parameter in the response.

d. The group member has performed a Version Request-Response exchange with the Group Manager (see Section 9.8). That is, V is different than V' specified by the 'num' parameter in the response.

In either case, the group member MUST delete the stored keying material with version number V.

If case (a) or case (b) applies, the group member MUST perform the following actions.

1. The group member MUST NOT install the latest keying material yet, in case that was already obtained.

2. The group member sends a Stale Sender IDs Request to the Group Manager (see Section 11.3.1), specifying the version number V as payload of the request.

   If the Stale Sender IDs Response from the Group Manager has no payload, the group member MUST remove all the authentication credentials from its list of group members' authentication credentials used in the group, and MUST delete all its Recipient Contexts used in the group.

   Otherwise, the group member considers the stale Sender IDs specified in the Stale Sender IDs Response from the Group Manager. Then, the group member MUST remove every authentication credential associated with a stale Sender ID from its list of group members' authentication credentials used in the group, and MUST delete each of its Recipient Contexts used in the group whose corresponding Recipient ID is a stale Sender ID.

3. The group member installs the latest keying material with version number V' and derives the corresponding new Security Context.

If case (c) or case (d) applies, the group member SHOULD perform the following actions.

1. The group member sends a Stale Sender IDs Request to the Group Manager (see Section 11.3.1), specifying the version number V as payload of the request.

   If the Stale Sender IDs Response from the Group Manager has no payload, the group member MUST remove all the authentication credentials from its list of group members' authentication credentials used in the group, and MUST delete all its Recipient Contexts used in the group.

   Otherwise, the group member considers the stale Sender IDs specified in the Stale Sender IDs Response from the Group Manager. Then, the group member MUST remove every authentication credential associated with a stale Sender ID from its list of group members' authentication credentials used in the group, and MUST delete each of its Recipient Contexts used in the group whose corresponding Recipient ID is a stale Sender ID.

2. The group member obtains the latest keying material with version number V' from the Group Manager. This can happen by sending a Key Distribution Request to the Group Manager (see Section 9.1.1) and Section 9.1.2).

3. The group member installs the latest keying material with version number V' and derives the corresponding new Security Context.

If case (c) or case (d) applies, the group member can alternatively perform the following actions.

1. The group member re-joins the group (see Section 6.1). When doing so, the group member MUST re-join with the same roles it currently has in the group, and MUST request from the Group Manager the authentication credentials of all the current group members. That is, the 'get_creds' parameter of the Join Request MUST be present and MUST be set to the CBOR simple value "null" (0xf6).

2. When receiving the Join Response (see Section 6.4 and Section 6.4), the group member retrieves the set Z of authentication credentials specified in the 'creds' parameter.

   Then, the group member MUST remove every authentication credential which is not in Z from its list of group members' authentication credentials used in the group, and MUST delete each of its Recipient Contexts used in the group that does not include any of the authentication credentials in Z.

3. The group member installs the latest keying material with version number V' and derives the corresponding new Security Context.

11.3.1. Retrieve Stale Sender IDs

When realizing to have missed one or more group rekeying instances (see Section 11.3), a node needs to retrieve from the Group Manager the data required to delete some of its stored group members' authentication credentials and Recipient Contexts (see Section 8.3.1). These data is provided as an aggregated set of stale Sender IDs, which are used as specified in Section 11.3.

That is, the node sends a CoAP FETCH request to the endpoint /ace-group/GROUPNAME/stale-sids at the Group Manager defined in Section 8.3 of this document, where GROUPNAME is the name of the OSCORE group.

The payload of the Stale Sender IDs Request MUST include a CBOR unsigned integer. This encodes the version number V of the most recent group keying material stored and installed by the requesting Client, which is older than the latest, possibly just distributed, keying material with version number V'.

The handler MUST reply with a 4.00 (Bad Request) error response, if the request is not formatted correctly. Also, the handler MUST respond with a 4.00 (Bad Request) error response, if the specified version number V is greater or equal than the version number V' associated with the latest keying material in the group, i.e., in case V >= V'.

Otherwise, the handler responds with a 2.05 (Content) Stale Sender IDs Response. The payload of the response is formatted as defined below, where SKEW = (V' - V + 1).

* The Group Manager considers ITEMS as the current number of sets of stale Sender IDs for the group (see Section 7.1).

* If SKEW > ITEMS, the Stale Sender IDs Response MUST NOT have a payload.

* Otherwise, the payload of the Stale Sender IDs Response MUST include a CBOR array, where each element is encoded as a CBOR byte string. The order of elements in the CBOR array is irrelevant. The Group Manager populates the CBOR array as follows.

  - The Group Manager creates an empty CBOR array ARRAY and an empty set X.

  - The Group Manager considers the SKEW most recent sets of stale Sender IDs for the group. Note that the most recent set is the one associated with the latest version of the group keying material.

  - The Group Manager copies all the Sender IDs from the selected sets into X. When doing so, the Group Manager MUST discard duplicates. That is, the same Sender ID MUST NOT be present more than once in the final content of X.

  - For each Sender ID in X, the Group Manager encodes it as a CBOR byte string and adds the result to ARRAY.

  - Finally, ARRAY is specified as payload of the Stale Sender IDs Response. Note that ARRAY might result in the empty CBOR array.

Figure 9 gives an overview of the exchange described above, while Figure 10 shows an example of Stale Sender IDs Request-Response.

Figure 9: Message Flow of Stale Sender IDs Request-Response

Request:

Header: FETCH (Code=0.05)
Uri-Host: "kdc.example.com"
Uri-Path: "ace-group"
Uri-Path: "g1"
Uri-Path: "stale-sids"
Payload (in CBOR diagnostic notation):
  42

Response:

Header: Content (Code=2.05)
Payload (in CBOR diagnostic notation):
  [h'01', h'fc', h'12ab', h'de44', h'ff']

Figure 10: Example of Stale Sender IDs Request-Response

12. ACE Groupcomm Parameters

In addition to those defined in Section 8 of [I-D.ietf-ace-key-groupcomm], this application profile defines additional parameters used during the second part of the message exchange with the Group Manager, i.e., after the exchange of Token Transfer Request and Response (see Section 5.3). The table below summarizes them and specifies the CBOR key to use instead of the full descriptive name.

Note that the media type application/ace-groupcomm+cbor MUST be used when these parameters are transported in the respective message fields.

Name	CBOR Key	CBOR Type	Reference
group_senderId	TBD	bstr	[RFC-XXXX]
ecdh_info	TBD	array	[RFC-XXXX]
kdc_dh_creds	TBD	array	[RFC-XXXX]
group_enc_key	TBD	bstr	[RFC-XXXX]
stale_node_ids	TBD	array	[RFC-XXXX]

Figure 11: ACE Groupcomm Parameters

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

The Group Manager is expected to support all the parameters above. Instead, a Client is required to support the new parameters defined in this application profile as specified below (REQ29).

* 'group_senderId' MUST be supported by a Client that intends to join an OSCORE group with the role of Requester and/or Responder.

* 'ecdh_info' MUST be supported by a Client that intends to join a group which uses the pairwise mode of Group OSCORE.

* 'kdc_dh_creds' MUST be supported by a Client that intends to join a group which uses the pairwise mode of Group OSCORE and that does not plan to or cannot rely on an early retrieval of the Group Manager's Diffie-Hellman authentication credential.

* 'group_enc_key' MUST be supported by a Client that intends to join a group which uses the group mode of Group OSCORE or to be signature verifier for that group.

* 'stale_node_ids' MUST be supported.

When the conditional parameters defined in Section 8 of [I-D.ietf-ace-key-groupcomm] are used with this application profile, a Client must, should or may support them as specified below (REQ30).

* 'client_cred', 'cnonce', 'client_cred_verify'. A Client that has an own authentication credential to use in a group MUST support these parameters.

* 'kdcchallenge'. A Client that has an own authentication credential to use in a group and that provides the Access Token to the Group Manager through a Token Transfer Request (see Section 5.3) MUST support this parameter.

* 'creds_repo'. This parameter is not relevant for this application profile, since the Group Manager always acts as repository of the group members' authentication credentials.

* 'group_policies'. A Client that is interested in the specific policies used in a group, but that does not know them or cannot become aware of them before joining that group, SHOULD support this parameter.

* 'peer_roles'. A Client MUST support this parameter, since in this application profile it is relevant for Clients to know the roles of the group member associated with each authentication credential.

* 'kdc_nonce', 'kdc_cred' and 'kdc_cred_verify'. A Client MUST support these parameters, since the Group Manager's authentication credential is required to process messages protected with Group OSCORE (see Section 4.3 of [I-D.ietf-core-oscore-groupcomm]).

* 'mgt_key_material'. A Client that supports an advanced rekeying scheme possibly used in the group, such as based on one-to-many rekeying messages sent by the Group Manager (e.g., over IP multicast), MUST support this parameter.

* 'control_group_uri'. A Client that supports the hosting of local resources each associated with a group (hence acting as CoAP server) and the reception of one-to-many requests sent to those resources by the Group Manager (e.g., over IP multicast) MUST support this parameter.

13. ACE Groupcomm Error Identifiers

In addition to those defined in Section 9 of [I-D.ietf-ace-key-groupcomm], this application profile defines new values that the Group Manager can include as error identifiers, in the 'error' field of an error response with Content-Format application/ace-groupcomm+cbor.

Value	Description
7	Signatures not used in the group
8	Operation permitted only to signature verifiers
9	Group currently not active

Figure 12: ACE Groupcomm Error Identifiers

A Client supporting the 'error' parameter (see Sections 4.1.2 and 8 of [I-D.ietf-ace-key-groupcomm]) and able to understand the specified error may use that information to determine what actions to take next. If it is included in the error response and supported by the Client, the 'error_description' parameter may provide additional context. The following guidelines apply.

* In case of error 7, the Client should stop sending the request in question to the Group Manager. In this application profile, this error is relevant only for a signature verifier, in case it tries to access resources related to a pairwise-only group.

* In case of error 8, the Client should stop sending the request in question to the Group Manager.

* In case of error 9, the Client should wait for a certain (pre-configured) amount of time, before trying to re-send its request to the Group Manager.

14. Default Values for Group Configuration Parameters

This section defines the default values that the Group Manager assumes for the configuration parameters of an OSCORE group, unless differently specified when creating and configuring the group. This can be achieved as specified in [I-D.ietf-ace-oscore-gm-admin].

A possible reason for the Group Manager to consider default values different from those recommended in this section is to ensure that each of those are consistent with what the Group Manager supports, e.g., in terms of signature algorithm and format of authentication credentials used in the OSCORE group.

This ensures that the Group Manager is able to perform the operations defined in this document, e.g., to achieve proof-of-possession of a joining node's private key (see Section 6.2), as well as to provide a joining node with its own authentication credential and the associated proof-of-possession challenge (see Section 6.3).

14.1. Common

This section always applies, as related to common configuration parameters.

* For the HKDF Algorithm 'hkdf', the Group Manager SHOULD use HKDF SHA-256, defined as default in Section 3.2 of [RFC8613]. In the 'hkdf' parameter, this HKDF Algorithm is specified by the HMAC Algorithm HMAC 256/256 (COSE algorithm encoding: 5).

* For the format 'cred_fmt' used for the authentication credentials in the group, the Group Manager SHOULD use CBOR Web Token (CWT) or CWT Claims Set (CCS) [RFC8392], i.e., the COSE Header Parameter 'kcwt' and 'kccs', respectively.

  [These COSE Header Parameters are under pending registration requested by draft-ietf-lake-edhoc.]

* For 'max_stale_sets', the Group Manager SHOULD consider N = 3 as the maximum number of stored sets of stale Sender IDs for the group (see Section 7.1).

14.2. Group Mode

This section applies if the group uses (also) the group mode of Group OSCORE.

* For the Signature Encryption Algorithm 'sign_enc_alg' used to encrypt messages protected with the group mode, the Group Manager SHOULD use AES-CCM-16-64-128 (COSE algorithm encoding: 10) as default value.

The Group Manager SHOULD use the following default values for the Signature Algorithm 'sign_alg' and related parameters 'sign_params', consistently with the "COSE Algorithms" registry [COSE.Algorithms], the "COSE Key Types" registry [COSE.Key.Types] and the "COSE Elliptic Curves" registry [COSE.Elliptic.Curves].

* For the Signature Algorithm 'sign_alg' used to sign messages protected with the group mode, the signature algorithm EdDSA [RFC8032].

* For the parameters 'sign_params' of the Signature Algorithm:

  - The array [[OKP], [OKP, Ed25519]], in case EdDSA is assumed or specified for 'sign_alg'. This indicates to use the COSE key type OKP and the elliptic curve Ed25519 [RFC8032].

  - The array [[EC2], [EC2, P-256]], in case ES256 [RFC6979] is specified for 'sign_alg'. This indicates to use the COSE key type EC2 and the elliptic curve P-256.

  - The array [[EC2], [EC2, P-384]], in case ES384 [RFC6979] is specified for 'sign_alg'. This indicates to use the COSE key type EC2 and the elliptic curve P-384.

  - The array [[EC2], [EC2, P-521]], in case ES512 [RFC6979] is specified for 'sign_alg'. This indicates to use the COSE key type EC2 and the elliptic curve P-521.

  - The array [[RSA], [RSA]], in case PS256, PS384 or PS512 [RFC8017] is specified for 'sign_alg'. This indicates to use the COSE key type RSA.

14.3. Pairwise Mode

This section applies if the group uses (also) the pairwise mode of Group OSCORE.

For the AEAD Algorithm 'alg' used to encrypt messages protected with the pairwise mode, the Group Manager SHOULD use the same default value defined in Section 3.2 of [RFC8613], i.e., AES-CCM-16-64-128 (COSE algorithm encoding: 10).

For the Pairwise Key Agreement Algorithm 'ecdh_alg' and related parameters 'ecdh_params', the Group Manager SHOULD use the following default values, consistently with the "COSE Algorithms" registry [COSE.Algorithms], the "COSE Key Types" registry [COSE.Key.Types] and the "COSE Elliptic Curves" registry [COSE.Elliptic.Curves].

* For the Pairwise Key Agreement Algorithm 'ecdh_alg' used to compute static-static Diffie-Hellman shared secrets, the ECDH algorithm ECDH-SS + HKDF-256 specified in Section 6.3.1 of [RFC9053].

* For the parameters 'ecdh_params' of the Pairwise Key Agreement Algorithm:

- The array [[OKP], [OKP, X25519]], in case EdDSA is assumed or specified for 'sign_alg', or in case the group is a pairwise-only group. This indicates to use the COSE key type OKP and the elliptic curve X25519 [RFC8032].

- The array [[EC2], [EC2, P-256]], in case ES256 [RFC6979] is specified for 'sign_alg'. This indicates to use the COSE key type EC2 and the elliptic curve P-256.

- The array [[EC2], [EC2, P-384]], in case ES384 [RFC6979] is specified for 'sign_alg'. This indicates to use the COSE key type EC2 and the elliptic curve P-384.

- The array [[EC2], [EC2, P-521]], in case ES512 [RFC6979] is specified for 'sign_alg'. This indicates to use the COSE key type EC2 and the elliptic curve P-521.

15. Security Considerations

Security considerations for this profile are inherited from [I-D.ietf-ace-key-groupcomm], the ACE framework for Authentication and Authorization [RFC9200], and the specific transport profile of ACE signalled by the AS, such as [RFC9202] and [RFC9203].

The following security considerations also apply for this profile.

15.1. Management of OSCORE Groups

This profile leverages the following management aspects related to OSCORE groups and discussed in the sections of [I-D.ietf-core-oscore-groupcomm] referred below.

* Management of group keying material (see Section 3.2 of [I-D.ietf-core-oscore-groupcomm]). The Group Manager is responsible for the renewal and re-distribution of the keying material in the groups of its competence (rekeying).

  The Group Manager performs a rekeying when one or more members leave the group, thus preserving forward security and ensuring that the security properties of Group OSCORE are fulfilled. According to the specific application requirements, the Group Manager can also rekey the group upon a new node's joining, in case backward security has also to be preserved.

* Provisioning and retrieval of authentication credentials (see Section 3 of [I-D.ietf-core-oscore-groupcomm]). The Group Manager acts as repository of authentication credentials of group members, and provides them upon request.

* Synchronization of sequence numbers (see Section 6.3 of [I-D.ietf-core-oscore-groupcomm]). This concerns how a responder node that has just joined an OSCORE group can synchronize with the sequence number of requesters in the same group.

Before sending the Join Response, the Group Manager MUST verify that the joining node actually owns the associated private key. To this end, the Group Manager can rely on the proof-of-possession challenge-response defined in Section 6.

Alternatively, when establishing a secure communication association with the Group Manager, the joining node can provide the Group Manager with its own authentication credential, and use the public key included thereof as asymmetric proof-of-possession key. For example, this is the case when the joining node relies on Section 3.2.2 of [RFC9202] and authenticates itself during the DTLS handshake with the Group Manager. However, this requires the authentication credential to be in the format used in the OSCORE group, and that both the authentication credential of the joining node and the included public key are compatible with the signature or ECDH algorithm, and possible associated parameters used in the OSCORE group.

A node may have joined multiple OSCORE groups under different non-synchronized Group Managers. Therefore, it can happen that those OSCORE groups have the same Group Identifier (Gid). It follows that, upon receiving a Group OSCORE message addressed to one of those groups, the node would have multiple Security Contexts matching with the Gid in the incoming message. It is up to the application to decide how to handle such collisions of Group Identifiers, e.g., by trying to process the incoming message using one Security Context at the time until the right one is found.

15.2. Size of Nonces as Proof-of-Possesion Challenge

With reference to the Join Request message in Section 6.1, the proof-of-possession (PoP) evidence included in 'client_cred_verify' is computed over an input including also N_C | N_S, where | denotes concatenation.

For the N_C challenge, it is RECOMMENDED to use an 8-byte long random nonce. Furthermore, N_C is always conveyed in the 'cnonce' parameter of the Join Request, which is always sent over the secure communication association between the joining node and the Group Manager.

As defined in Section 6.1.1, the way the N_S value is computed depends on the particular way the joining node provides the Group Manager with the Access Token, as well as on following interactions between the two.

* If the Access Token has not been provided to the Group Manager by means of a Token Transfer Request to the /authz-info endpoint as in Section 5.3, then N_S is computed as a 32-byte long challenge. For an example, see point (2) of Section 6.1.1.

* If the Access Token has been provided to the Group Manager by means of a Token Transfer Request to the /authz-info endpoint as in Section 5.3, then N_S takes the most recent value provided to the Client by the Group Manager in the 'kdcchallenge' parameter, as specified in point (1) of Section 6.1.1. This value is provided either in the Token Transfer Response (see Section 5.3), or in a 4.00 (Bad Request) error response to a following Join Request (see Section 6.2). In either case, it is RECOMMENDED to use an 8-byte long random challenge as value for N_S.

If we consider both N_C and N_S to take 8-byte long values, the following considerations hold.

* Let us consider both N_C and N_S as taking random values, and the Group Manager to never change the value of the N_S provided to a Client during the lifetime of an Access Token. Then, as per the birthday paradox, the average collision for N_S will happen after 2^32 new transferred Access Tokens, while the average collision for N_C will happen after 2^32 new Join Requests. This amounts to considerably more token provisionings than the expected new joinings to OSCORE groups under a same Group Manager, as well as to considerably more requests to join OSCORE groups from a same Client using a same Access Token under a same Group Manager.

* Section 7 of [RFC9203] as well Appendix B.2 of [RFC8613] recommend the use of 8-byte random values as well. Unlike in those cases, the values of N_C and N_S considered in this document are not used for as sensitive operations as the derivation of a Security Context, and thus do not have possible implications in the security of AEAD ciphers.

15.3. Reusage of Nonces for Proof-of-Possession Input

As long as the Group Manager preserves the same N_S value currently associated with an Access Token, i.e., the latest value provided to a Client in a 'kdcchallenge' parameter, the Client is able to successfully reuse the same proof-of-possession (PoP) input for multiple Join Requests to that Group Manager.

In particular, the Client can reuse the same N_C value for every Join Request to the Group Manager, and combine it with the same unchanged N_S value. This results in reusing the same PoP input for producing the PoP evidence to include in the 'client_cred_verify' parameter of the Join Requests.

Unless the Group Manager maintains a list of N_C values already used by that Client since the latest update to the N_S value associated with the Access Token, the Group Manager can be forced to falsely believe that the Client possesses its own private key at that point in time, upon verifying the PoP evidence in the 'client_cred_verify' parameter.

16. 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.

16.1. OAuth Parameters

IANA is asked to register the following entries to the "OAuth Parameters" registry, as per the procedure specified in Section 11.2 of [RFC6749].

* Parameter name: ecdh_info

* Parameter usage location: client-rs request, rs-client response

* Change Controller: IESG

* Specification Document(s): [RFC-XXXX]

* Parameter name: kdc_dh_creds

* Parameter usage location: client-rs request, rs-client response

* Change Controller: IESG

* Specification Document(s): [RFC-XXXX]

16.2. OAuth Parameters CBOR Mappings

IANA is asked to register the following entries to the "OAuth Parameters CBOR Mappings" registry, as per the procedure specified in Section 8.10 of [RFC9200].

* Name: ecdh_info

* CBOR Key: TBD (range -256 to 255)

* Value Type: Simple value "null" / Array

* Reference: [RFC-XXXX]

* Name: kdc_dh_creds

* CBOR Key: TBD (range -256 to 255)

* Value Type: Simple value "null" / Array

* Reference: [RFC-XXXX]

16.3. ACE Groupcomm Parameters

IANA is asked to register the following entries to the "ACE Groupcomm Parameters" registry defined in Section 11.6 of [I-D.ietf-ace-key-groupcomm].

* Name: group_senderId

* CBOR Key: TBD

* CBOR Type: Byte string

* Reference: [RFC-XXXX] (Section 9.2)

* Name: ecdh_info

* CBOR Key: TBD

* CBOR Type: Array

* Reference: [RFC-XXXX] (Section 6.2)

* Name: kdc_dh_creds

* CBOR Key: TBD

* CBOR Type: Array

* Reference: [RFC-XXXX] (Section 6.2)

* Name: group_enc_key

* CBOR Key: TBD

* CBOR Type: Byte string

* Reference: [RFC-XXXX] (Section 8.2.1)

* Name: stale_node_ids

* CBOR Key: TBD

* CBOR Type: Array

* Reference: [RFC-XXXX] (Section 11)

16.4. ACE Groupcomm Key Types

IANA is asked to register the following entry to the "ACE Groupcomm Key Types" registry defined in Section 11.7 of [I-D.ietf-ace-key-groupcomm].

* Name: Group_OSCORE_Input_Material object

* Key Type Value: GROUPCOMM_KEY_TBD

* Profile: "coap_group_oscore_app", defined in Section 16.5 of this document.

* Description: A Group_OSCORE_Input_Material object encoded as described in Section 6.3 of this document.

* Reference: [RFC-XXXX] (Section 6.3)

16.5. ACE Groupcomm Profiles

IANA is asked to register the following entry to the "ACE Groupcomm Profiles" registry defined in Section 11.8 of [I-D.ietf-ace-key-groupcomm].

* Name: coap_group_oscore_app

* Description: Application profile to provision keying material for participating in group communication protected with Group OSCORE as per [I-D.ietf-core-oscore-groupcomm].

* CBOR Value: PROFILE_TBD

* Reference: [RFC-XXXX] (Section 6.3)

16.6. OSCORE Security Context Parameters

IANA is asked to register the following entries in the "OSCORE Security Context Parameters" registry defined in Section 9.4 of [RFC9203].

* Name: group_SenderId
* CBOR Label: TBD
* CBOR Type: Byte string
* Registry: -
* Description: OSCORE Sender ID assigned to a member of an OSCORE group
* Reference: [RFC-XXXX] (Section 6.3)

* Name: cred_fmt
* CBOR Label: TBD
* CBOR Type: Integer
* Registry: COSE Header Parameters
* Description: Format of authentication credentials to be used in the OSCORE group
* Reference: [RFC-XXXX] (Section 6.3)

* Name: sign_enc_alg
* CBOR Label: TBD
* CBOR Type: Text string / Integer
* Registry: COSE Algorithms
* Description: OSCORE Signature Encryption Algorithm Value
* Reference: [RFC-XXXX] (Section 6.3)

   * Name: sign_alg

   * CBOR Label: TBD

   * CBOR Type: Text string / Integer

   * Registry: COSE Algorithms

   * Description: OSCORE Signature Algorithm Value

   * Reference: [RFC-XXXX] (Section 6.3)

   * Name: sign_params

   * CBOR Label: TBD

   * CBOR Type: Array

   * Registry: COSE Algorithms, COSE Key Types, COSE Elliptic Curves

   * Description: OSCORE Signature Algorithm Parameters

   * Reference: [RFC-XXXX] (Section 6.3)

   * Name: ecdh_alg

   * CBOR Label: TBD

   * CBOR Type: Text string / Integer

   * Registry: COSE Algorithms

   * Description: OSCORE Pairwise Key Agreement Algorithm Value

   * Reference: [RFC-XXXX] (Section 6.3)

   * Name: ecdh_params

   * CBOR Label: TBD

   * CBOR Type: Array

   * Registry: COSE Algorithms, COSE Key Types, COSE Elliptic Curves

   * Description: OSCORE Pairwise Key Agreement Algorithm Parameters

* Reference: [RFC-XXXX] (Section 6.3)

16.7. TLS Exporter Labels

IANA is asked to register the following entry to the "TLS Exporter Labels" registry defined in Section 6 of [RFC5705] and updated in Section 12 of [RFC8447].

* Value: EXPORTER-ACE-Sign-Challenge-coap-group-oscore-app

* DTLS-OK: Y

* Recommended: N

* Reference: [RFC-XXXX] (Section 6.1.1)

16.8. AIF Media-Type Sub-Parameters

For the media-types application/aif+cbor and application/aif+json defined in Section 5.1 of [RFC9237], IANA is requested to register the following entries for the two media-type parameters Toid and Tperm, in the respective sub-registry defined in Section 5.2 of [RFC9237] within the "MIME Media Type Sub-Parameter" registry group.

* Parameter: Toid

* Name: oscore-gname

* Description/Specification: OSCORE group name

* Reference: [RFC-XXXX]

* Parameter: Tperm

* Name: oscore-gperm

* Description/Specification: permissions pertaining OSCORE groups

* Reference: [RFC-XXXX]

16.9. CoAP Content-Format

IANA is asked to register the following entries to the "CoAP Content-Formats" registry within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

* Media Type: application/aif+cbor;Toid="oscore-gname",Tperm="oscore-gperm"

* Encoding: -

* ID: 292 (suggested)

* Reference: [RFC-XXXX]

* Media Type: application/aif+json;Toid="oscore-gname",Tperm="oscore-gperm"

* Encoding: -

* ID: 293 (suggested)

* Reference: [RFC-XXXX]

16.10. Group OSCORE Roles

This document establishes the IANA "Group OSCORE Roles" registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 16.13.

This registry includes the possible roles that nodes can take in an OSCORE group, each in combination with a numeric identifier. These numeric identifiers are used to express authorization information about joining OSCORE groups, as specified in Section 3 of [RFC-XXXX].

The columns of this registry are:

* Name: A value that can be used in documents for easier comprehension, to identify a possible role that nodes can take in an OSCORE group.

* Value: The numeric identifier for this role. Integer values greater than 65535 are marked as "Private Use", all other values use the registration policy "Expert Review" [RFC8126].

* Description: This field contains a brief description of the role.

* Reference: This contains a pointer to the public specification for the role.

This registry will be initially populated by the values in Figure 1.

The Reference column for all of these entries will be [RFC-XXXX].

16.11. CoRE Resource Type

IANA is asked to register the following entry in the "Resource Type (rt=) Link Target Attribute Values" registry within the "Constrained Restful Environments (CoRE) Parameters" registry group.

* Value: "core.osc.gm"

* Description: Group-membership resource of an OSCORE Group Manager.

* Reference: [RFC-XXXX]

Client applications can use this resource type to discover a group membership resource at an OSCORE Group Manager, where to send a request for joining the corresponding OSCORE group.

16.12. ACE Groupcomm Errors

IANA is asked to register the following entries in the "ACE Groupcomm Errors" registry defined in Section 11.11 of [I-D.ietf-ace-key-groupcomm].

* Value: 7

* Description: Signatures not used in the group.

* Reference: [RFC-XXXX]

* Value: 8

* Description: Operation permitted only to signature verifiers.

* Reference: [RFC-XXXX]

* Value: 9

* Description: Group currently not active.

* Reference: [RFC-XXXX]

16.13. Expert Review Instructions

The IANA registry established in this document is defined as "Expert Review". This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

* Clarity and correctness of registrations. Experts are expected to check the clarity of purpose and use of the requested entries. Experts should inspect the entry for the considered role, to verify the correctness of its description against the role as intended in the specification that defined it. Experts should consider requesting an opinion on the correctness of registered parameters from the Authentication and Authorization for Constrained Environments (ACE) Working Group and the Constrained RESTful Environments (CoRE) Working Group.

  Entries that do not meet these objectives of clarity and completeness should not be registered.

* Duplicated registration and point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments.

* Experts should take into account the expected usage of roles when approving point assignments. Given a 'Value' V as code point, the length of the encoding of (2^(V+1) - 1) should be weighed against the usage of the entry, considering the resources and capabilities of devices it will be used on. Additionally, given a 'Value' V as code point, the length of the encoding of (2^(V+1) - 1) should be weighed against how many code points resulting in that encoding length are left, and the resources and capabilities of devices it will be used on.

* Specifications are recommended. When specifications are not provided, the description provided needs to have sufficient information to verify the points above.

17. References

17.1. Normative References

[COSE.Algorithms]
IANA, "COSE Algorithms", <https://www.iana.org/assignments/cose/cose.xhtml#algorithms>.

[COSE.Elliptic.Curves]
IANA, "COSE Elliptic Curves", <https://www.iana.org/assignments/cose/cose.xhtml#elliptic-curves>.

[COSE.Header.Parameters]
IANA, "COSE Header Parameters", <https://www.iana.org/assignments/cose/cose.xhtml#header-parameters>.

[COSE.Key.Types]
IANA, "COSE Key Types", <https://www.iana.org/assignments/cose/cose.xhtml#key-type>.

[I-D.ietf-ace-key-groupcomm]
Palombini, F. and M. Tiloca, "Key Provisioning for Group Communication using ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-16, 5 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-16>.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., Mattsson, J. P., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-17, 20 December 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-17>.

[NIST-800-56A]
Barker, E., Chen, L., Roginsky, A., Vassilev, A., and R. Davis, "Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography - NIST Special Publication 800-56A, Revision 3", April 2018, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf>.

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

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

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

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

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

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

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

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

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/info/rfc8447>.

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

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

[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/info/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/info/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/info/rfc9053>.

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

[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/info/rfc9200>.

[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/info/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/info/rfc9203>.

[RFC9237] Bormann, C., "An Authorization Information Format (AIF) for Authentication and Authorization for Constrained Environments (ACE)", RFC 9237, DOI 10.17487/RFC9237, August 2022, <https://www.rfc-editor.org/info/rfc9237>.

[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/info/rfc9277>.

17.2. Informative References

[I-D.ietf-ace-oscore-gm-admin]
Tiloca, M., Höglund, R., Van der Stok, P., and F. Palombini, "Admin Interface for the OSCORE Group Manager", Work in Progress, Internet-Draft, draft-ietf-ace-oscore-gm-admin-07, 24 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-oscore-gm-admin-07>.

[I-D.ietf-core-coap-pubsub]
Koster, M., Keränen, A., and J. Jimenez, "Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-coap-pubsub-11, 7 November 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-core-coap-pubsub-11>.

[I-D.ietf-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-08, 11 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-08>.

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-05, 10 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-05>.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsüss, C., and P. Van der Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-discovery-12, 5 September 2022, <https://datatracker.ietf.org/doc/html/draft-tiloca-core-oscore-discovery-12>.

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

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/info/rfc6690>.

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

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/info/rfc7925>.

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

Appendix A. Profile Requirements

This section lists how this application profile of ACE addresses the requirements defined in Appendix A of [I-D.ietf-ace-key-groupcomm].

A.1. Mandatory-to-Address Requirements

* REQ1 - Specify the format and encoding of 'scope'. This includes defining the set of possible roles and their identifiers, as well as the corresponding encoding to use in the scope entries according to the used scope format: see Section 3 and Section 5.1.

* REQ2 - If the AIF format of 'scope' is used, register its specific instance of "Toid" and "Tperm" as Media Type parameters and a corresponding Content-Format, as per the guidelines in [RFC9237]: see Section 16.8 and Section 16.9.

* REQ3 - if used, specify the acceptable values for 'sign_alg': values from the "Value" column of the "COSE Algorithms" registry [COSE.Algorithms].

* REQ4 - If used, specify the acceptable values for 'sign_parameters': format and values from the COSE algorithm capabilities as specified in the "COSE Algorithms" registry [COSE.Algorithms].

* REQ5 - If used, specify the acceptable values for 'sign_key_parameters': format and values from the COSE key type capabilities as specified in the "COSE Key Types" registry [COSE.Key.Types].

* REQ6 - Specify the acceptable formats for authentication credentials and, if used, the acceptable values for 'cred_fmt': acceptable formats explicitly provide the public key as well as the comprehensive set of information related to the public key algorithm (see Section 5.3 and Section 6.3). Consistent acceptable values for 'cred_fmt' are taken from the "Label" column of the "COSE Header Parameters" registry [COSE.Header.Parameters].

* REQ7 - If the value of the GROUPNAME URI path and the group name in the Access Token scope (gname in Section 3.1 of [I-D.ietf-ace-key-groupcomm]) are not required to coincide, specify the mechanism to map the GROUPNAME value in the URI to the group name: not applicable, since a perfect matching is required.

* REQ8 - Define whether the KDC has an authentication credential and if this has to be provided through the 'kdc_cred' parameter, see Section 4.1 of [I-D.ietf-ace-key-groupcomm]: yes, as required by the Group OSCORE protocol [I-D.ietf-core-oscore-groupcomm], see Section 6.3 of this document.

* REQ9 - Specify if any part of the KDC interface as defined in Section 4.1 of [I-D.ietf-ace-key-groupcomm] is not supported by the KDC: not applicable.

* REQ10 - Register a Resource Type for the root url-path, which is used to discover the correct url to access at the KDC (see Section 4.1 of [I-D.ietf-ace-key-groupcomm]): the Resource Type (rt=) Link Target Attribute value "core.osc.gm" is registered in Section 16.11.

* REQ11 - Define what specific actions (e.g., CoAP methods) are allowed on each resource provided by the KDC interface, depending on whether the Client is a current group member; the roles that a Client is authorized to take as per the obtained access token; and the roles that the Client has as current group member: see Section 8.4.

* REQ12 - Categorize possible newly defined operations for Clients into primary operations expected to be minimally supported and secondary operations, and provide accompanying considerations: see Section 8.5.

* REQ13 - Specify the encoding of group identifier (see Section 4.2.1 of [I-D.ietf-ace-key-groupcomm]): CBOR byte string (see Section 9.10).

* REQ14 - Specify the approaches used to compute and verify the PoP evidence to include in 'client_cred_verify', and which of those approaches is used in which case: see Section 6.1 and Section 6.2.

* REQ15 - Specify how the nonce N_S is generated, if the token is not provided to the KDC through the Token Transfer Request to the authz-info endpoint (e.g., if it is used directly to validate TLS instead): see Section 6.1.1.

* REQ16 - Define the initial value of the 'num' parameter: the initial value MUST be set to 0 when creating the OSCORE group, e.g., as in [I-D.ietf-ace-oscore-gm-admin].

* REQ17 - Specify the format of the 'key' parameter: see Section 6.3.

* REQ18 - Specify acceptable values of the 'gkty' parameter: Group_OSCORE_Input_Material object (see Section 6.3).

* REQ19 - Specify and register the application profile identifier: coap_group_oscore_app (see Section 16.5).

* REQ20 - If used, specify the format and content of 'group_policies' and its entries: see Section 6.3.

* REQ21 - Specify the approaches used to compute and verify the PoP evidence to include in 'kdc_cred_verify', and which of those approaches is used in which case: see Section 6.3, Section 6.4 and Section 9.5.

* REQ22 - Specify the communication protocol that the members of the group must use: CoAP [RFC7252], also for group communication [I-D.ietf-core-groupcomm-bis].

* REQ23 - Specify the security protocols that the group members must use to protect their communication: Group OSCORE [I-D.ietf-core-oscore-groupcomm].

* REQ24 - Specify how the communication is secured between the Client and KDC: by means of any transport profile of ACE [RFC9200] between Client and Group Manager that complies with the requirements in Appendix C of [RFC9200].

* REQ25 - Specify the format of the identifiers of group members: the Sender ID used in the OSCORE group (see Section 6.3 and Section 9.3).

* REQ26 - Specify policies at the KDC to handle member ids that are not included in 'get_creds': see Section 9.3.

* REQ27 - Specify the format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label: see Section 9.2.

* REQ28 - Specify which CBOR tag is used for identifying the semantics of binary scopes, or register a new CBOR tag if a suitable one does not exist already: see Section 5.2.

* REQ29 - Categorize newly defined parameters according to the same criteria of Section 8 of [I-D.ietf-ace-key-groupcomm]: see Section 12.

* REQ30 - Define whether Clients must, should or may support the conditional parameters defined in Section 8 of [I-D.ietf-ace-key-groupcomm], and under which circumstances: see Section 12.

A.2. Optional-to-Address Requirements

* OPT1 (Optional) – If the textual format of 'scope' is used, specify CBOR values to use for abbreviating the role identifiers in the group: not applicable.

* OPT2 (Optional) – Specify additional parameters used in the exchange of Token Transfer Request and Response:

  - 'ecdh_info', to negotiate the ECDH algorithm, ECDH algorithm parameters, ECDH key parameters and exact format of authentication credentials used in the group, in case the joining node supports the pairwise mode of Group OSCORE (see Section 5.3).

  - 'kdc_dh_creds', to ask for and retrieve the Group Manager's Diffie-Hellman authentication credentials, in case the joining node supports the pairwise mode of Group OSCORE and the Access Token authorizes to join parwise-only groups (see Section 5.3).

* OPT3 (Optional) – Specify the negotiation of parameter values for signature algorithm and signature keys, if 'sign_info' is not used: possible early discovery by using the approach based on the CoRE Resource Directory described in [I-D.tiloca-core-oscore-discovery].

* OPT4 (Optional) – Specify possible or required payload formats for specific error cases: send a 4.00 (Bad Request) error response to a Join Request (see Section 6.2).

* OPT5 (Optional) – Specify additional identifiers of error types, as values of the 'error' field in an error response from the KDC: see Section 16.12.

* OPT6 (Optional) – Specify the encoding of 'creds_repo' if the default is not used: no.

* OPT7 (Optional) – Specify the functionalities implemented at the 'control_uri' resource hosted at the Client, including message exchange encoding and other details (see Section 4.3.1 of [I-D.ietf-ace-key-groupcomm]): see Section 10 for the eviction of a group member; see Section 11 for the group rekeying process.

* OPT8 (Optional) – Specify the behavior of the handler in case of failure to retrieve an authentication credential for the specific node: send a 4.00 (Bad Request) error response to a Join Request (see Section 6.2).

* OPT9 (Optional) - Define a default group rekeying scheme, to refer to in case the 'rekeying_scheme' parameter is not included in the Join Response (see Section 4.3.1.1 of [I-D.ietf-ace-key-groupcomm]): the "Point-to-Point" rekeying scheme registered in Section 11.12 of [I-D.ietf-ace-key-groupcomm], whose detailed use for this profile is defined in Section 11 of this document.

* OPT10 (Optional) - Specify the functionalities implemented at the 'control_group_uri' resource hosted at the Client, including message exchange encoding and other details (see Section 4.3.1 of [I-D.ietf-ace-key-groupcomm]): see Section 10 for the eviction of multiple group members.

* OPT11 (Optional) - Specify policies that instruct Clients to retain unsuccessfully decrypted messages and for how long, so that they can be decrypted after getting updated keying material: no.

* OPT12 (Optional) - Specify for the KDC to perform group rekeying (together or instead of renewing individual keying material) when receiving a Key Renewal Request: the Group Manager SHOULD NOT perform a group rekeying, unless already scheduled to occur shortly (see Section 9.2).

* OPT13 (Optional) - Specify how the identifier of a group members's authentication credential is included in requests sent to other group members: no.

* OPT14 (Optional) - Specify additional information to include in rekeying messages for the "Point-to-Point" group rekeying scheme (see Section 6.1 of [I-D.ietf-ace-key-groupcomm]): see Section 11.1.

* OPT15 (Optional) - Specify if Clients must or should support any of the parameters defined as optional in Section 8 of [I-D.ietf-ace-key-groupcomm]: no.

Appendix B. Extensibility for Future COSE Algorithms

As defined in Section 8.1 of [RFC9053], future algorithms can be registered in the "COSE Algorithms" registry [COSE.Algorithms] as specifying none or multiple COSE capabilities.

To enable the seamless use of such future registered algorithms, this section defines a general, agile format for:

* Each 'ecdh_info_entry' of the 'ecdh_info' parameter in the Token Transfer Response (see Section 5.3 and Section 5.3.1);

* The 'sign_params' and 'ecdh_params' parameters within the 'key' parameter (see Section 6.3), as part of the response payloads used in Section 6.3, Section 9.1.1, Section 9.1.2 and Section 11.

Appendix B of [I-D.ietf-ace-key-groupcomm] describes the analogous general format for 'sign_info_entry' of the 'sign_info' parameter in the Token Transfer Response (see Section 5.3 of this document).

If any of the currently registered COSE algorithms is considered, using this general format yields the same structure defined in this document for the items above, thus ensuring retro-compatibility.

B.1. Format of 'ecdh_info_entry'

The format of each 'ecdh_info_entry' (see Section 5.3 and Section 5.3.1) is generalized as follows. Given N the number of elements of the 'ecdh_parameters' array, i.e., the number of COSE capabilities of the ECDH algorithm, then:

* 'ecdh_key_parameters' is replaced by N elements 'ecdh_capab_i', each of which is a CBOR array.

* The i-th array following 'ecdh_parameters', i.e., 'ecdh_capab_i' (i = 0, ..., N-1), is the array of COSE capabilities for the algorithm capability specified in 'ecdh_parameters'[i].

The CDDL notation [RFC8610] of the 'ecdh_info_entry' parameter is given below.

```
ecdh_info_entry =
[
  id : gname / [ + gname ],
  ecdh_alg : int / tstr,
  ecdh_parameters : [ alg_capab_1 : any,
                      alg_capab_2 : any,
                      ...,
                      alg_capab_N : any],
  ecdh_capab_1 : [ any ],
  ecdh_capab_2 : [ any ],
  ...,
  ecdh_capab_N : [ any ],
  cred_fmt : int / null
]

gname = tstr
```

Figure 13: 'ecdh_info_entry' with general format

B.2. Format of 'key'

The format of 'key' (see Section 6.3) is generalized as follows.

* The 'sign_params' array includes N+1 elements, whose exact structure and value depend on the value of the signature algorithm specified in 'sign_alg'.

  - The first element, i.e., 'sign_params'[0], is the array of the N COSE capabilities for the signature algorithm, as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms] (see Section 8.1 of [RFC9053]).

  - Each following element 'sign_params'[i], i.e., with index i > 0, is the array of COSE capabilities for the algorithm capability specified in 'sign_params'[0][i-1].

  For example, if 'sign_params'[0][0] specifies the key type as capability of the algorithm, then 'sign_params'[1] is the array of COSE capabilities for the COSE key type associated with the signature algorithm, as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types] (see Section 8.2 of [RFC9053]).

* The 'ecdh_params' array includes M+1 elements, whose exact structure and value depend on the value of the ECDH algorithm specified in 'ecdh_alg'.

  - The first element, i.e., 'ecdh_params'[0], is the array of the M COSE capabilities for the ECDH algorithm, as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [COSE.Algorithms] (see Section 8.1 of [RFC9053]).

  - Each following element 'ecdh_params'[i], i.e., with index i > 0, is the array of COSE capabilities for the algorithm capability specified in 'ecdh_params'[0][i-1].

  For example, if 'ecdh_params'[0][0] specifies the key type as capability of the algorithm, then 'ecdh_params'[1] is the array of COSE capabilities for the COSE key type associated with the ECDH algorithm, as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [COSE.Key.Types] (see Section 8.2 of [RFC9053]).

Appendix C. Document Updates

RFC EDITOR: PLEASE REMOVE THIS SECTION.

C.1. Version -15 to -16

* Early mentioning of invalid combinations of roles.
* Revised presentation of handling of stale Sender IDs.
* Fixed CDDL notation.
* Fixed diagnostic notation in examples.
* Possible reason to deviate from default parameter values.
* Clarifications and editorial fixes.

C.2. Version -14 to -15

* Alignment with renaming in draft-ietf-ace-key-groupcomm.
* Updated signaling of semantics for binary encoded scopes.
* Considered the upload of Access Tokens in the DTLS 1.3 Handshake.
* Fixes in IANA registrations.
* Editorial fixes.

C.3. Version -13 to -14

* Major reordering of the document sections.
* The HKDF Algorithm is specified by the HMAC Algorithm.
* Group communication does not necessarily use IP multicast.
* Generalized AIF data model, also for draft-ace-oscore-gm-admin.
* Clarifications and editorial improvements.

C.4. Version -12 to -13

* Renamed parameters about authentication credentials.
* It is optional for the Group Manager to reassign Gids by tracking "Birth Gids".

* Distinction between authentication credentials and public keys.

* Updated IANA considerations related to AIF.

* Updated textual description of registered ACE Scope Semantics value.

C.5. Version -11 to -12

* Clarified semantics of 'ecdh_info' and 'kdc_dh_creds'.

* Definition of /ace-group/GROUPNAME/kdc-pub-key moved to draft-ietf-ace-key-groupcomm.

* ace-group/ accessible also to non-members that are not Verifiers.

* Clarified what resources are accessible to Verifiers.

* Revised error handling for the newly defined resources.

* Revised use of CoAP error codes.

* Use of "Token Tranfer Request" and "Token Transfer Response".

* New parameter 'rekeying_scheme'.

* Categorization of new parameters and inherited conditional parameters.

* Clarifications on what to do in case of enhanced error responses.

* Changed UCCS to CCS.

* Authentication credentials of just joined Clients can be in rekeying messages.

* Revised names of new IANA registries.

* Clarified meaning of registered CoRE resource type.

* Alignment to new requirements from draft-ietf-ace-key-groupcomm.

* Fixes and editorial improvements.

C.6. Version -10 to -11

* Removed redundancy of key type capabilities, from 'sign_info', 'ecdh_info' and 'key'.

* New resource to retrieve the Group Manager's authentication credential.

* New resource to retrieve material for Signature Verifiers.

* New parameter 'sign_enc_alg' related to the group mode.

* 'cred_fmt' takes value from the COSE Header Parameters registry.

* Improved alignment of the Join Response payload with the Group OSCORE Security Context parameters.

* Recycling Group IDs by tracking "Birth GIDs".

* Error handling in case of non available Sender IDs upon joining.

* Error handling in case EdDSA public keys with invalid Y coordinate when the pairwise mode of Group OSCORE is supported.

* Generalized proof-of-possession (PoP) for the joining node's private key; defined Diffie-Hellman based PoP for OSCORE groups using only the pairwise mode.

* Proof-of-possession of the Group Manager's private key in the Join Response.

* Always use 'peer_identifiers' to convey Sender IDs as node identifiers.

* Stale Sender IDs provided when rekeying the group.

* New resource for late retrieval of stale Sender IDs.

* Added examples of message exchanges.

* Revised default values of group configuration parameters.

* Fixes to IANA registrations.

* General format of parameters related to COSE capabilities, supporting future registered COSE algorithms (new Appendix).

C.7. Version -09 to -10

* Updated non-recycling policy of Sender IDs.

* Removed policies about Sender Sequence Number synchronization.

* 'control_path' renamed to 'control_uri'.
* Format of 'get_pub_keys' aligned with draft-ietf-ace-key-groupcomm.
* Additional way to inform of group eviction.
* Registered semantics identifier for extended scope format.
* Extended error handling, with error type specified in some error responses.
* Renumbered requirements.

C.8. Version -08 to -09

* The url-path "ace-group" is used.
* Added overview of admitted methods on the Group Manager resources.
* Added exchange of parameters relevant for the pairwise mode of Group OSCORE.
* The signed value for 'client_cred_verify' includes also the scope.
* Renamed the key material object as Group_OSCORE_Input_Material object.
* Replaced 'clientId' with 'group_SenderId'.
* Added message exchange for Group Names request-response.
* No reassignment of Sender ID and Gid in the same OSCORE group.
* Updates on group rekeying contextual with request of new Sender ID.
* Signature verifiers can also retrieve Group Names and URIs.
* Removed group policy about supporting Group OSCORE in pairwise mode.
* Registration of the resource type rt="core.osc.gm".
* Update list of requirements.
* Clarifications and editorial revision.

C.9. Version -07 to -08

* AIF specific data model to express scope entries.
* A set of roles is checked as valid when processing the Join Request.
* Updated format of 'get_pub_keys' in the Join Request.
* Payload format and default values of group policies in the Join Response.
* Updated payload format of the FETCH request to retrieve authentication credentials.
* Default values for group configuration parameters.
* IANA registrations to support the AIF specific data model.

C.10. Version -06 to -07

* Alignments with draft-ietf-core-oscore-groupcomm.
* New format of 'sign_info', using the COSE capabilities.
* New format of Join Response parameters, using the COSE capabilities.
* Considerations on group rekeying.
* Editorial revision.

C.11. Version -05 to -06

* Added role of external signature verifier.
* Parameter 'rsnonce' renamed to 'kdcchallenge'.
* Parameter 'kdcchallenge' may be omitted in some cases.
* Clarified difference between group name and OSCORE Gid.
* Removed the role combination ["requester", "monitor"].
* Admit implicit scope and audience in the Authorization Request.
* New format for the 'sign_info' parameter.

* Scope not mandatory to include in the Join Request.

* Group policy about supporting Group OSCORE in pairwise mode.

* Possible individual rekeying of a single requesting node combined with a group rekeying.

* Security considerations on reusage of signature challenges.

* Addressing optional requirement OPT12 from draft-ietf-ace-key-groupcomm

* Editorial improvements.

C.12. Version -04 to -05

* Nonce N_S also in error responses to the Join Requests.

* Supporting single Access Token for multiple groups/topics.

* Supporting legal requesters/responders using the 'peer_roles' parameter.

* Registered and used dedicated label for TLS Exporter.

* Added method for uploading a new authentication credential to the Group Manager.

* Added resource and method for retrieving the current group status.

* Fixed inconsistency in retrieving group keying material only.

* Clarified retrieval of keying material for monitor-only members.

* Clarification on incrementing version number when rekeying the group.

* Clarification on what is re-distributed with the group rekeying.

* Security considerations on the size of the nonces used for the signature challenge.

* Added CBOR values to abbreviate role identifiers in the group.

C.13. Version -03 to -04

* New abstract.

* Moved general content to draft-ietf-ace-key-groupcomm
* Terminology: node name; node resource.
* Creation and pointing at node resource.
* Updated Group Manager API (REST methods and offered services).
* Size of challenges 'cnonce' and 'rsnonce'.
* Value of 'rsnonce' for reused or non-traditionally-posted tokens.
* Removed reference to RFC 7390.
* New requirements from draft-ietf-ace-key-groupcomm
* Editorial improvements.

C.14. Version -02 to -03

* New sections, aligned with the interface of ace-key-groupcomm .
* Exchange of information on the signature algorithm and related parameters, during the Token POST (Section 4.1).
* Nonce 'rsnonce' from the Group Manager to the Client (Section 4.1).
* Client PoP signature in the Key Distribution Request upon joining (Section 4.2).
* Local actions on the Group Manager, upon a new node's joining (Section 4.2).
* Local actions on the Group Manager, upon a node's leaving (Section 12).
* IANA registration in ACE Groupcomm Parameters registry.
* More fulfilled profile requirements (Appendix A).

C.15. Version -01 to -02

* Editorial fixes.
* Changed: "listener" to "responder"; "pure listener" to "monitor".

* Changed profile name to "coap_group_oscore_app", to reflect it is an application profile.

* Added the 'type' parameter for all requests to a Join Resource.

* Added parameters to indicate the encoding of authentication credentials.

* Challenge-response for proof-of-possession of signature keys (Section 4).

* Renamed 'key_info' parameter to 'sign_info'; updated its format; extended to include also parameters of the signature key (Section 4.1).

* Code 4.00 (Bad request), in responses to joining nodes providing an invalid authentication credential (Section 4.3).

* Clarifications on provisioning and checking of authentication credentials (Sections 4 and 6).

* Extended discussion on group rekeying and possible different approaches (Section 7).

* Extended security considerations: proof-of-possession of signature keys; collision of OSCORE Group Identifiers (Section 8).

* Registered three entries in the IANA registry "Sequence Number Synchronization Method" (Section 9).

* Registered one public key encoding in the "ACE Public Key Encoding" IANA registry (Section 9).

C.16. Version -00 to -01

* Changed name of 'req_aud' to 'audience' in the Authorization Request (Section 3.1).

* Added negotiation of signature algorithm/parameters between Client and Group Manager (Section 4).

* Updated format of the Key Distribution Response as a whole (Section 4.3).

* Added parameter 'cs_params' in the 'key' parameter of the Key Distribution Response (Section 4.3).

* New IANA registrations in the "ACE Authorization Server Request Creation Hints" registry, "ACE Groupcomm Key" registry, "OSCORE Security Context Parameters" registry and "ACE Groupcomm Profile" registry (Section 9).

Acknowledgments

The authors sincerely thank Christian Amsüss, Santiago Aragón, Stefan Beck, Carsten Bormann, Martin Gunnarsson, Rikard Höglund, Watson Ladd, Daniel Migault, Jim Schaad, Ludwig Seitz, Göran Selander and Peter van der Stok for their comments and feedback.

The work on this document has been partly supported by VINNOVA and the Celtic-Next project CRITISEC; by the H2020 project SIFIS-Home (Grant agreement 952652); and by the EIT-Digital High Impact Initiative ACTIVE.

Authors' Addresses

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-164 29 Stockholm Kista
Sweden
Email: marco.tiloca@ri.se

Jiye Park
Universitaet Duisburg-Essen
Schuetzenbahn 70
45127 Essen
Germany
Email: ji-ye.park@uni-due.de

Francesca Palombini
Ericsson AB
Torshamnsgatan 23
SE-16440 Stockholm Kista
Sweden
Email: francesca.palombini@ericsson.com

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2024

M. Tiloca
R. Höglund
RISE AB
P. van der Stok

F. Palombini
Ericsson AB

4 March 2024

Admin Interface for the OSCORE Group Manager
draft-ietf-ace-oscore-gm-admin-11

Abstract

Group communication for CoAP can be secured using Group Object Security for Constrained RESTful Environments (Group OSCORE). A Group Manager is responsible for handling the joining of new group members, as well as managing and distributing the group keying material. This document defines a RESTful admin interface at the Group Manager that allows an Administrator entity to create and delete OSCORE groups, as well as to retrieve and update their configuration. The ACE framework for Authentication and Authorization is used to enforce authentication and authorization of the Administrator at the Group Manager. Protocol-specific transport profiles of ACE are used to achieve communication security, proof-of-possession, and server authentication.

Discussion Venues

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

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 5 September 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 4
  1.1. Terminology 5
2. Group Administration 7
  2.1. Managing OSCORE Groups 8
  2.2. Collection Representation 9
  2.3. Discovery 9
3. Format of Scope 9
  3.1. On Using Group Name Patterns 14
4. Getting Access to the Group Manager 14
  4.1. Multiple Administrators for the Same OSCORE Group 18
5. Group Configurations 19
  5.1. Group Configuration Representation 19
    5.1.1. Configuration Properties 19
    5.1.2. Status Properties 22
  5.2. Default Values 23
    5.2.1. Configuration Parameters 23
    5.2.2. Status Parameters 24
6. Interactions with the Group Manager 25
  6.1. Retrieve the Full List of Group Configurations 26
  6.2. Retrieve a List of Group Configurations by Filters . . . 26

6.3. Create a New Group Configuration 29
6.4. Retrieve a Group Configuration 35
6.5. Retrieve Part of a Group Configuration by Filters 37
6.6. Overwrite a Group Configuration 39
6.6.1. Effects on Joining Nodes 43
6.6.2. Effects on the Group Members 43
6.7. Selective Update of a Group Configuration 47
6.7.1. Effects on Joining Nodes 51
6.7.2. Effects on the Group Members 51
6.8. Delete a Group Configuration 51
6.8.1. Effects on the Group Members 53
7. ACE Groupcomm Parameters 54
8. ACE Groupcomm Error Identifiers 56
9. Security Considerations 58
9.1. Change of Group Configuration 59
9.2. Group Manager 60
9.3. Administrators 60
10. IANA Considerations 61
10.1. ACE Groupcomm Parameters 61
10.2. ACE Groupcomm Errors 63
10.3. Resource Types 64
10.4. Group OSCORE Admin Permissions 64
10.5. Expert Review Instructions 65
11. References 66
11.1. Normative References 66
11.2. Informative References 69
Appendix A. Processing of Group Name Patterns at the AS 70
Appendix B. Document Updates 72
B.1. Version -10 to -11 72
B.2. Version -09 to -10 73
B.3. Version -08 to -09 73
B.4. Version -07 to -08 73
B.5. Version -06 to -07 74
B.6. Version -05 to -06 74
B.7. Version -04 to -05 75
B.8. Version -03 to -04 75
B.9. Version -02 to -03 75
B.10. Version -01 to -02 76
B.11. Version -00 to -01 76
Acknowledgments 76
Authors' Addresses 76

1. Introduction

The Constrained Application Protocol (CoAP) [RFC7252] can also be used for group communication [I-D.ietf-core-groupcomm-bis], where messages are exchanged between members of a group, e.g., over IP multicast. Applications relying on CoAP can achieve end-to-end security at the application layer by using Object Security for Constrained RESTful Environments (OSCORE) [RFC8613], and especially Group OSCORE [I-D.ietf-core-oscore-groupcomm] in group communication scenarios.

When group communication for CoAP is protected with Group OSCORE, nodes are required to join the correct OSCORE group explicitly. To this end, a joining node interacts with a Group Manager (GM) entity responsible for that group, and retrieves the required keying material to securely communicate with other group members using Group OSCORE.

The method in [I-D.ietf-ace-key-groupcomm-oscore] specifies how nodes can join an OSCORE group through the respective Group Manager. That method builds on the ACE framework for Authentication and Authorization [RFC9200], ensuring a secure joining process as well as authentication and authorization of joining nodes (clients) at the Group Manager (resource server).

In some deployments, the application running on the Group Manager may know when a new OSCORE group has to be created, as well as how it should be configured and later on updated or deleted, e.g., based on the current application state or pre-installed policies. In this case, the Group Manager application can create and configure OSCORE groups when needed, by using a local application interface. However, this requires the Group Manager to be application-specific, which in turn may lead to error-prone deployments and is poorly flexible.

In other deployments, a separate Administrator entity, such as a Commissioning Tool, is directly responsible for creating and configuring the OSCORE groups at a Group Manager, as well as for maintaining them during their whole lifetime until their deletion. This allows the Group Manager to be agnostic of the specific applications using secure group communication.

This document specifies a RESTful admin interface at the Group Manager, intended for an Administrator as a separate entity external to the Group Manager and its application. The interface allows the Administrator to create and delete OSCORE groups, as well as to specify and update their configuration.

Interaction examples are provided in Link Format [RFC6690] and in CBOR [RFC8949]. The examples in CBOR are expressed in CBOR diagnostic notation without the tag and value abbreviations.

The ACE framework is used to ensure authentication and authorization of the Administrator (client) at the Group Manager (resource server). In order to achieve communication security, proof-of-possession, and server authentication, the Administrator and the Group Manager leverage protocol-specific transport profiles of ACE, such as [RFC9202][RFC9203]. These include also possible forthcoming transport profiles that comply with the requirements in Appendix C of [RFC9200].

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 from the following specifications.

* CDDL [RFC8610], CBOR [RFC8949] and COSE [RFC9052][RFC9053].

* The CoAP protocol [RFC7252], also in group communication scenarios [I-D.ietf-core-groupcomm-bis]. These especially include the following concepts.

  - "application group", as a set of CoAP nodes that share a common set of resources.

  - "security group", as a set of CoAP nodes that share the same security material, and use it to protect and verify exchanged messages.

* The OSCORE [RFC8613] and Group OSCORE [I-D.ietf-core-oscore-groupcomm] security protocols. These especially include the following concepts.

  - Group Manager, as the entity responsible for a set of OSCORE groups where communications among members are secured using Group OSCORE. An OSCORE group is used as a security group for one or many application groups.

  - Authentication credential, as the set of information associated with an entity, including that entity's public key and parameters associated with the public key. Examples of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC5280], and C509 certificates [I-D.ietf-cose-cbor-encoded-cert].

* The ACE framework for authentication and authorization [RFC9200]. 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).

* The management of keying material for groups in ACE [I-D.ietf-ace-key-groupcomm] and specifically for OSCORE groups [I-D.ietf-ace-key-groupcomm-oscore]. These include the concept of group-membership resource hosted by the Group Manager that new members access to join the OSCORE group, and that current members can access to retrieve updated keying material.

Note that the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol".

This document also refers to the following terminology:

* Administrator: entity responsible for creating, configuring, and deleting OSCORE groups at a Group Manager.

* Group name: stable and invariant name of an OSCORE group. The group name MUST be unique under the same Group Manager, and MUST include only characters that are valid for a URI path segment.

* Group-collection resource: a single-instance resource hosted by the Group Manager. An Administrator accesses a group-collection resource to retrieve the list of existing OSCORE groups, or to create a new OSCORE group, under that Group Manager.

  As an example, this document uses /manage as the url-path of the group-collection resource; implementations are not required to use this name, and can define their own instead.

* Group-configuration resource: a resource hosted by the Group Manager, associated with an OSCORE group under that Group Manager. A group-configuration resource is identifiable with the invariant group name of the respective OSCORE group. An Administrator accesses a group-configuration resource to retrieve or change the configuration of the respective OSCORE group, or to delete that group.

  The url-path to a group-configuration resource has GROUPNAME as last segment, with GROUPNAME the invariant group name assigned upon its creation. Building on the considered url-path of the group-collection resource, this document uses /manage/GROUPNAME as the url-path of a group-configuration resource; implementations are not required to use this same construct, and can define their own instead.

* Admin resource: a group-collection resource or a group-configuration resource hosted by the Group Manager.

2. Group Administration

With reference to the ACE framework and the terminology defined in OAuth 2.0 [RFC6749]:

* The Group Manager acts as Resource Server (RS). It provides one single group-collection resource, and one group-configuration resource per existing OSCORE group.

* The Administrator acts as Client (C), and requests to access the group-collection resource and group-configuration resources at the Group Manager.

* The Authorization Server (AS) authorizes the Administrator to access the group-collection resource and group-configuration resources at a Group Manager. Multiple Group Managers can be associated with the same AS.

  The authorized access for an Administrator can be limited to performing only a subset of operations, according to what is allowed by the authorization information in the Access Token issued to that Administrator (see Section 3 and Section 4). The AS can authorize multiple Administrators to access the group-collection resource and the (same) group-configuration resources at the Group Manager.

  The AS MAY issue Access Tokens to the Administrator for other purposes than accessing admin resources of registered Group Managers. For example, an Access Token can specify authorization

information for joining OSCORE groups at a Group Manager (see [I-D.ietf-ace-key-groupcomm-oscore]), possibly combined with authorization information for accessing admin resources at the same Group Manager (see Section 3). Also, an AS can of course issue an Access Token that specifies authorization information unrelated to OSCORE groups, but instead pertaining to the access of other resources hosted by the Group Manager or other Resource Servers.

2.1. Managing OSCORE Groups

Figure 1 shows the resources of a Group Manager available to an Administrator.

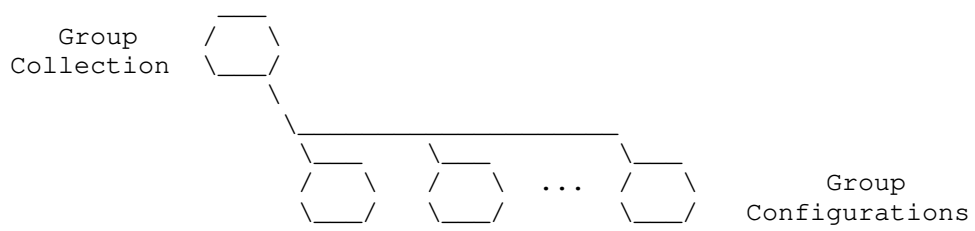


Figure 1: Admin Resources of a Group Manager

The Group Manager exports a single group-collection resource, with resource type "core.osc.gcoll" registered in Section 10.3 of this document. The interface for the group-collection resource defined in Section 6 allows the Administrator to:

* Retrieve the list of existing OSCORE groups.

* Retrieve a list of existing OSCORE groups matching with specified filter criteria.

* Create a new OSCORE group, specifying its invariant group name and, optionally, its configuration.

The Group Manager exports one group-configuration resource for each of its OSCORE groups. Each group-configuration resource has resource type "core.osc.gconf" registered in Section 10.3 of this document, and is identified by the group name specified upon creating the OSCORE group. The interface for a group-configuration resource defined in Section 6 allows the Administrator to:

* Retrieve the complete current configuration of the OSCORE group.

* Retrieve part of the current configuration of the OSCORE group, by applying filter criteria.

* Overwrite the current configuration of the OSCORE group.

* Selectively update only part of the current configuration of the OSCORE group.

* Delete the OSCORE group.

2.2. Collection Representation

A collection of group configurations is represented as a Link Format document [RFC6690] containing the list of corresponding group-configuration resources.

Each group configuration is represented as a link, which specifies the URI of the group-configuration resource as link target, and the link target attribute 'rt' (Resource Type) with value "core.osc.gconf".

2.3. Discovery

The Administrator can discover group-collection resources from a Resource Directory [RFC9176] or from .well-known/core, by using the resource type "core.osc.gcoll".

The Administrator can discover group-configuration resources for the group-collection resource as specified in Section 6.1 and Section 6.2.

3. Format of Scope

This section defines the exact format and encoding of scope to use, in order to express authorization information for the Administrator (see Section 4).

To this end, this document uses the Authorization Information Format (AIF) [RFC9237]. In particular, it uses and extends the AIF specific data model AIF-OSCORE-GROUPCOMM defined in Section 3 of [I-D.ietf-ace-key-groupcomm-oscore].

The original definition of the data model AIF-OSCORE-GROUPCOMM specifies a scope as structured into scope entries, which express authorization information for users of an OSCORE group, i.e., actual group members or external signature verifiers. Hereafter, these are referred to as "user scope entries".

This document extends the same AIF specific data model AIF-OSCORE-GROUPCOMM as defined below. In particular, it defines how the same scope can (also) include scope entries that express authorization information for Administrators of OSCORE groups. Hereafter, these are referred to as "admin scope entries", or simply as "scope entries" unless otherwise indicated.

Like in the original definition of the data model AIF-OSCORE-GROUPCOMM, and with reference to the generic AIF model

```
AIF-Generic<Toid, Tperm> = [* [Toid, Tperm]]
```

the value of the CBOR byte string used as scope encodes the CBOR array [* [Toid, Tperm]], where each [Toid, Tperm] element corresponds to one scope entry.

Then, the following applies for each admin scope entry intended to express authorization information for an Administrator, as defined in this document.

* The object identifier ("Toid") is specialized as either of the following, and specifies a group name pattern P for the admin scope entry.

  - Wildcard pattern: "Toid" is specialized as the CBOR simple value true (0xf5), specifying the wildcard pattern. That is, any group name expressed as a literal text string matches with this group name pattern.

  - Literal pattern: "Toid" is specialized as a CBOR text string, whose value specifies an exact group name as a literal string. That is, only one specific group name expressed as a literal text string matches with this group name pattern.

  - Complex pattern: "Toid" is specialized as a tagged CBOR data item, specifying a more complex group name pattern with the semantics signaled by the CBOR tag. That is, multiple group names expressed as a literal text string match with this group name pattern.

    For example, and as typically expected, the data item can be a CBOR text string marked with the CBOR tag 21065 or 35. This indicates that the group name pattern specified as value of the CBOR text string is a regular expression (see [RFC9485] and Section 3.4.5.3 of [RFC8949]).

In case the AIF specific data model AIF-OSCORE-GROUPCOMM is used in a JSON payload, the semantics information conveyed by the CBOR tag can be equivalently conveyed, for example, in a nested JSON object.

The AS and the Group Manager are expected to have agreed on commonly supported semantics for group name patterns. This can happen, for instance, as part of the registration process of the Group Manager at the AS.

* The permission set ("Tperm") is specialized as a CBOR unsigned integer with value Q. This specifies the permissions that the Administrator has with respect to performing operations on the admin resources at the Group Manager, as pertaining to any OSCORE group whose name matches with the pattern P. The value Q is computed as follows.

  - Each permission in the permission set is converted into the corresponding numeric identifier X from the "Value" column of the "Group OSCORE Admin Permissions" registry, for which this document defines the entries in Table 1.

  - The set of N numbers is converted into the single value Q, by taking two to the power of each numeric identifier X_1, X_2, ..., X_N, and then computing the inclusive OR of the binary representations of all the power values.

  In general, a single permission can be associated with multiple different operations that are possible to be performed when interacting with the Group Manager. For example, the "List" permission allows the Administrator to retrieve a list of group configurations (see Section 6.1) or only a subset of that according to specified filter criteria (see Section 6.2), by issuing a GET or FETCH request to the group-collection resource, respectively.

Name	Value	Description
List	0	Retrieve list of group configurations
Create	1	Create new group configurations
Read	2	Retrieve group configurations
Write	3	Change group configurations
Delete	4	Delete group configurations

Table 1: Numeric identifier of permissions on the admin resources at a Group Manager

The following CDDL [RFC8610] notation defines an admin scope entry that uses the data model AIF-OSCORE-GROUPCOMM and expresses a set of permissions from those in Table 1.

```
;# include rfc9237

AIF-OSCORE-GROUPCOMM = AIF-Generic<oscore-gname, oscore-gperm>

oscore-gname = true / tstr / #6.<uint>(any) ; Group name pattern
oscore-gperm = uint .bits admin-permissions

admin-permissions = &(
   List: 0,
   Create: 1,
   Read: 2,
   Write: 3,
   Delete: 4
)

scope_entry = [oscore-gname, oscore-gperm]
```

The following example in CBOR diagnostic notation shows a CBOR array including five scope entries as its elements.

```
[[true,   / Toid (wildcard pattern) /
  5       / Tperm (List + Read) /
 ],
 ["gp1",  / Toid (literal pattern) /
  31      / Tperm (List + Create +
            Read + Write + Delete) /
 ],
 ["gp2",  / Toid (literal pattern) /
  31      / Tperm (List + Create +
            Read + Write + Delete) /
 ],
 [21065("^[A-K][0-9][-a-z0-9]*$"),  / Toid (complex pattern) /
  31                                / Tperm (List + Create +
                                      Read + Write + Delete) /
 ],
 [21065("^[J-Z][0-9][-a-z0-9]*$"),  / Toid (complex pattern) /
  13                                / Tperm (List + Read + Write) /
 ]
]
```

Future specifications that define new permissions on the admin resources at the Group Manager MUST register a corresponding numeric identifier in the "Group OSCORE Admin Permissions" registry defined in Section 10.4 of this document.

When using the scope format as defined in this section, the permission set ("Tperm") of each admin scope entry MUST include the "List" permission. It follows that, when expressing permissions for Administrators of OSCORE groups as defined in this document, an admin scope entry has the least significant bit of "Tperm" always set to 1.

Therefore, an Administrator is always allowed to retrieve a list of existing group configurations. The exact elements included in the returned list are determined by the Group Manager, based on the group name patterns specified in the admin scope entries of the Administrator's Access Token, as well as on possible filter criteria specified in the request from the Administrator (see Section 6.1 and Section 6.2).

Building on the above, the same single scope can include user scope entries as well as admin scope entries, whose specific format is defined in Section 3 of [I-D.ietf-ace-key-groupcomm-oscore] and earlier in this section, respectively. The two types of scope entries can be unambiguously distinguished by means of the least significant bit of their permission set "Tperm", which has value 0 for the user scope entries and 1 for the admin scope entries.

The coexistence of user scope entries and admin scope entries within the same scope makes it possible to issue a single Access Token, in case the requesting Client wishes to be a user for some OSCORE groups and at the same time Administrator for some (other) OSCORE groups under the same Group Manager.

3.1. On Using Group Name Patterns

Having the object identifier ("Toid") specialized as a pattern confers a number of advantages.

* When relying on wildcard patterns and complex patterns, the encoded scope can be compact in size while allowing the Administrator to operate on large pools of group names.

* When relying on wildcard patterns and complex patterns, the Administrator and the AS do not need to know the exact group names for requesting and issuing an Access Token, respectively (see Section 4). In turn, the Group Manager can effectively take the final decision about the name to assign to an OSCORE group, upon its creation (see Section 6.3).

4. Getting Access to the Group Manager

All communications between the involved entities rely on the CoAP protocol and MUST be secured.

In particular, communications between the Administrator and the Group Manager leverage protocol-specific transport profiles of ACE to achieve communication security, proof-of-possession, and server authentication. To this end, the AS may explicitly signal the specific transport profile to use, consistently with requirements and assumptions defined in the ACE framework [RFC9200].

With reference to the AS, communications between the Administrator and the AS (/token endpoint) as well as between the Group Manager and the AS (/introspect endpoint) can be secured by different means, for instance using DTLS [RFC9147] or OSCORE [RFC8613]. Further details on how the AS secures communications (with the Administrator and the Group Manager) depend on the transport profile of ACE in use, and are out of the scope of this document.

In order to specify authorization information for Administrators, the format and encoding of scope defined in Section 3 of this document MUST be used, for both the 'scope' claim in the Access Token and the 'scope' parameter in the Authorization Request and Authorization Response exchanged with the AS (see Sections 5.8.1 and 5.8.2 of [RFC9200]).

Furthermore, the AS MAY use the extended format of scope defined in Section 7 of [I-D.ietf-ace-key-groupcomm] for the 'scope' claim of the Access Token. In such a case, the AS MUST use the CBOR tag with tag number TAG_NUMBER, associated with the CoAP Content-Format CF_ID for the media type application/aif+cbor registered in Section 16.9 of [I-D.ietf-ace-key-groupcomm-oscore].

Note to RFC Editor: In the previous paragraph, please replace "TAG_NUMBER" with the CBOR tag number computed as TN(ct) in Section 4.3 of [RFC9277], where ct is the ID assigned to the CoAP Content-Format CF_ID registered in Section 16.9 of [I-D.ietf-ace-key-groupcomm-oscore]. Then, please replace "CF_ID" with the ID assigned to that CoAP Content-Format. Finally, please delete this paragraph.

This indicates that the binary encoded scope, as conveying the actual access control information, follows the scope semantics of the AIF specific data model AIF-OSCORE-GROUPCOMM defined in Section 3 of [I-D.ietf-ace-key-groupcomm-oscore] and extended as per Section 3 of this document.

In order to get access to the Group Manager for managing OSCORE groups, an Administrator performs the following steps.

1. The Administrator requests an Access Token from the AS, in order to access the group-collection and group-configuration resources on the Group Manager. To this end, the Administrator sends to the AS an Authorization Request as defined in Section 5.8.1 of [RFC9200].

   If the 'scope' parameter in the Authorization Request includes scope entries whose "Toid" specifies a complex pattern (see Section 3), then all such scope entries MUST adhere to the same pattern semantics.

   The Administrator will start or continue using a secure communication association with the Group Manager, according to the response from the AS and the transport profile of ACE in use.

2. The AS processes the Authorization Request as defined in Section 5.8.2 of [RFC9200], especially verifying that the Administrator is authorized to obtain the requested permissions, or possibly a subset of those.

   The AS specifies the information on the authorization granted to the Administrator as the value of the 'scope' claim to include in the Access Token, in accordance with the scope format specified in Section 3. It is implementation specific which particular

approach the AS takes to evaluate the requested permissions against the access policies pertaining to the Administrator for the Group Manager in question. Appendix A provides an example of such an approach that the AS can use.

If the 'scope' parameter in the Authorization Request includes scope entries whose "Toid" specifies a complex pattern adhering to a certain pattern semantics, then that semantics MUST be used for all the scope entries in the 'scope' claim that specify a complex pattern.

The AS MUST include the 'scope' parameter in the Authorization Response defined in Section 5.8.2 of [RFC9200], when the value included in the Access Token differs from the one specified by the Administrator in the Authorization Request. In such a case, scope specifies the set of permissions that the Administrator actually has, with respect to performing operations at the Group Manager, encoded as specified in Section 3.

If the 'scope' parameter in the Authorization Request includes scope entries whose "Toid" specifies a complex pattern and any of the following conditions holds, then the AS MUST reply with a 4.00 (Bad Request) error response (see Section 5.8.3 of [RFC9200]). The 'error_description' parameter carried out in the response payload MUST specify the CBOR value 1 (invalid_scope).

* The "Toid" of the different scope entries that specify a complex pattern do not all adhere to the same pattern semantics.

* The "Toid" of the different scope entries that specify a complex pattern adhere to the same pattern semantics, but this is not supported by the AS or by the Group Manager.

Finally, as discussed in Section 3, the authorization information included in the Authorization Request or specified by the AS might also include permissions for the same Client as a user of an OSCORE group, i.e., as an actual group member or an external signature verifier. As per Section 3, such authorization information is expressed by "user scope entries", whose format and processing is specified in [I-D.ietf-ace-key-groupcomm-oscore].

3. The Administrator transfers authentication and authorization information to the Group Manager by posting the obtained Access Token, according to the used profile of ACE, such as [RFC9202] and [RFC9203]. After that, the Administrator must have a secure communication association established with the Group Manager,

before performing any administrative operation on that Group Manager. Possible ways to provide secure communication are DTLS [RFC9147] and OSCORE [RFC8613]. The Administrator and the Group Manager maintain the secure association, to support possible future communications.

4. Consistently with what is allowed by the authorization information in the Access Token, the Administrator performs administrative operations at the Group Manager, as described in Section 6. These include retrieving a list of existing OSCORE groups, creating new OSCORE groups, retrieving and changing OSCORE group configurations, and removing OSCORE groups. Messages exchanged among the Administrator and the Group Manager are specified in Section 6.

   Upon receiving a request from the Administrator targeting the group-collection resource or a group-configuration resource, the Group Manager MUST check that it is storing a valid Access Token for that Administrator. If this is not the case, the Group Manager MUST reply with a 4.01 (Unauthorized) error response.

   If the request targets the group-collection resource for creating a new group with name GROUPNAME or it targets the group-configuration resource associated with an existing group with name GROUPNAME, then the Group Manager MUST check that it is storing a valid Access Token from that Administrator, such that the 'scope' claim specified in the Access Token: i) expresses authorization information through scope entries as defined in Section 3; and ii) specifically includes a scope entry where:

   * The group name GROUPNAME matches with the pattern specified by the "Toid" of the scope entry; and

   * The permission set specified by the "Tperm" of the scope entry allows the Administrator to perform the requested administrative operation on the targeted group-configuration resource.

   Note that the checks defined above only consider scope entries expressing permissions for administrative operations, namely "admin scope entries" as defined in Section 3, while the alternative "user scope entries" defined in [I-D.ietf-ace-key-groupcomm-oscore] are not considered.

   Further detailed checks to perform are defined separately for each operation at the Group Manager, when specified in Section 6.

In case the Group Manager stores a valid Access Token but the verifications above fail, the Group Manager MUST reply with a 4.03 (Forbidden) error response. This response MAY be an AS Request Creation Hints, as defined in Section 5.3 of [RFC9200], in which case the Content-Format MUST be set to application/ace+cbor (i.e., ID: 19).

If the request is not formatted correctly (e.g., required fields are not present or are not encoded as expected), the Group Manager MUST reply with a 4.00 (Bad Request) error response.

4.1. Multiple Administrators for the Same OSCORE Group

In addition to a "main" primary Administrator responsible for an OSCORE group at the Group Manager, it is also possible to have "assistant" secondary Administrators that are effectively authorized to perform some operations on the same OSCORE group.

With respect to the main Administrator, such assistant Administrators are expected to have fewer permissions to perform administrative operations related to the OSCORE group at the Group Manager. For example, they may not be authorized to create an OSCORE group, or to delete an OSCORE group and its configuration.

In case the main Administrator of an OSCORE group is dismissed or relinquishes its role, one of the assistant Administrators can be "promoted" and become main Administrator for that OSCORE group. Practically, this requires that the access policies associated with the promoted Administrator are updated accordingly at the Authorization Server. Also, the promoted Administrator has to request from the Authorization Server a new Access Token and to upload it to the Group Manager. If allowed by the used transport profile of ACE, this process can efficiently enforce a dynamic update of access rights, thus preserving the current secure association between the promoted Administrator and the Group Manager.

If an Administrator is not sure about being the only Administrator responsible for an OSCORE group, then it is RECOMMENDED that the Administrator ensures to have a recent representation of the group-configuration resource associated with the OSCORE group before overwriting (see Section 6.6), updating (see Section 6.7), or deleting (see Section 6.8) the group configuration. This can be achieved in the following ways.

* The Administrator performs a regular polling of the group configuration, by sending a GET request to the corresponding group-configuration resource (see Section 6.4).

* If the group-configuration resource associated with the OSCORE group is Observable, then the Administrator subscribes to that resource by using CoAP Observe [RFC7641]. The Observation request is a GET request sent to the group-configuration resource (see Section 6.4). In such a case, the Group Manager will also send a 4.04 (Not Found) response in case another Administrator deletes the group-configuration resource, as a result of deleting the associated OSCORE group and its configuration.

If the Administrator gains knowledge that the group configuration has changed compared to the latest known representation, then the Administrator might hold the execution of writing or deletion operation on the group-configuration resource, and first attempt checking with other Administrators responsible for the same OSCORE group about the changes they have made.

5. Group Configurations

A group configuration consists of a set of parameters.

5.1. Group Configuration Representation

The group configuration representation is a CBOR map, which includes configuration properties and status properties.

5.1.1. Configuration Properties

The CBOR map includes the following configuration parameters, whose CBOR abbreviations are defined in Section 7 of this document.

* 'hkdf', which specifies the HKDF Algorithm used in the OSCORE group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]), encoded as a CBOR text string or a CBOR integer. Possible values are the same ones admitted for the 'hkdf' parameter of the Group_OSCORE_Input_Material object, defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

* 'cred_fmt', which specifies the Authentication Credential Format used in the OSCORE group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]), encoded as a CBOR integer. Possible values are the same ones admitted for the 'cred_fmt' parameter of the Group_OSCORE_Input_Material object, defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

* 'group_mode', encoded as a CBOR simple value. Its value is true (0xf5) if the OSCORE group uses the group mode of Group OSCORE (see Section 8 of [I-D.ietf-core-oscore-groupcomm]), or false (0xf4) otherwise.

* 'gp_enc_alg', which is formatted as follows. If the configuration parameter 'group_mode' has value false (0xf4), this parameter has as value the CBOR simple value null (0xf6). Otherwise, this parameter specifies the Group Encryption Algorithm used in the OSCORE group to encrypt messages protected with the group mode (see Section 2 of [I-D.ietf-core-oscore-groupcomm]), encoded as a CBOR text string or a CBOR integer. Possible values are the same ones admitted for the 'sign_enc_alg' parameter of the Group_OSCORE_Input_Material object, defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

  Editor's note: as per the text above, the referred version of [I-D.ietf-ace-key-groupcomm-oscore] still uses 'sign_enc_alg' as parameter name. The next version of [I-D.ietf-ace-key-groupcomm-oscore] will be updated in order to use 'gp_enc_alg' instead, as already done for this document and consistently with the naming used in the latest version of [I-D.ietf-core-oscore-groupcomm].

* 'sign_alg', which is formatted as follows. If the configuration parameter 'group_mode' has value false (0xf4), this parameter has as value the CBOR simple value null (0xf6). Otherwise, this parameter specifies the Signature Algorithm used in the OSCORE group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]), encoded as a CBOR text string or a CBOR integer. Possible values are the same ones admitted for the 'sign_alg' parameter of the Group_OSCORE_Input_Material object, defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

* 'sign_params', which is formatted as follows. If the configuration parameter 'group_mode' has value false (0xf4), this parameter has as value the CBOR simple value null (0xf6). Otherwise, this parameter specifies the additional parameters for the Signature Algorithm used in the OSCORE group, encoded as a CBOR array. Possible formats and values are the same ones admitted for the 'sign_params' parameter of the Group_OSCORE_Input_Material object, defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

* 'pairwise_mode', encoded as a CBOR simple value. Its value is true (0xf5) if the OSCORE group uses the pairwise mode of Group OSCORE (see Section 9 of [I-D.ietf-core-oscore-groupcomm]), or false (0xf4) otherwise.

* 'alg', which is formatted as follows. If the configuration parameter 'pairwise_mode' has value false (0xf4), this parameter has as value the CBOR simple value null (0xf6). Otherwise, this parameter specifies the AEAD Algorithm used in the OSCORE group to

encrypt messages protected with the pairwise mode (see Section 2 of [I-D.ietf-core-oscore-groupcomm]), encoded as a CBOR text string or a CBOR integer. Possible values are the same ones admitted for the 'alg' parameter of the Group_OSCORE_Input_Material object, defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

* 'ecdh_alg', which is formatted as follows. If the configuration parameter 'pairwise_mode' has value false (0xf4), this parameter has as value the CBOR simple value null (0xf6). Otherwise, this parameter specifies the Pairwise Key Agreement Algorithm used in the OSCORE group (see Section 2 of [I-D.ietf-core-oscore-groupcomm]), encoded as a CBOR text string or a CBOR integer. Possible values are the same ones admitted for the 'ecdh_alg' parameter of the Group_OSCORE_Input_Material object, defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

* 'ecdh_params', which is formatted as follows. If the configuration parameter 'pairwise_mode' has value false (0xf4), this parameter has as value the CBOR simple value null (0xf6). Otherwise, this parameter specifies the parameters for the Pairwise Key Agreement Algorithm used in the OSCORE group, encoded as a CBOR array. Possible formats and values are the same ones admitted for the 'ecdh_params' parameter of the Group_OSCORE_Input_Material object, defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

* 'det_req', encoded as a CBOR simple value. Its value is true (0xf5) if the OSCORE group uses deterministic requests as defined in [I-D.amsuess-core-cachable-oscore], or false (0xf4) otherwise. This parameter MUST NOT be present if the configuration parameter 'group_mode' has value false (0xf4).

* 'det_hash_alg', encoded as a CBOR integer or text string. If present, this parameter specifies the Hash Algorithm used in the OSCORE group when producing deterministic requests, as defined in [I-D.amsuess-core-cachable-oscore]. This parameter takes values from the "Value" column of the "COSE Algorithms" Registry [COSE.Algorithms].

  This parameter MUST NOT be present if the configuration parameter 'det_req' is not present or if it is present with value false (0xf4). If the configuration parameter 'det_req' is present with value true (0xf5) and 'det_hash_alg' is not present, the choice of the Hash Algorithm to use when producing deterministic requests is left to the Group Manager.

5.1.2. Status Properties

The CBOR map includes the following status parameters. Unless specified otherwise, these are defined in this document and their CBOR abbreviations are defined in Section 7.

* 'rt', with value the resource type "core.osc.gconf" associated with group-configuration resources, encoded as a CBOR text string.

* 'active', encoding the CBOR simple value true (0xf5) if the OSCORE group is currently active, or the CBOR simple value false (0xf4) otherwise.

* 'group_name', with value the group name of the OSCORE group encoded as a CBOR text string.

* 'group_description', with value either a human-readable description of the OSCORE group encoded as a CBOR text string, or the CBOR simple value null (0xf6) if no description is specified.

* 'ace_groupcomm_profile', defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], with value "coap_group_oscore_app" defined in Section 16.5 of [I-D.ietf-ace-key-groupcomm-oscore] encoded as a CBOR integer.

* 'max_stale_sets', encoding a CBOR unsigned integer with value strictly greater than 1. With reference to Section 7.1 of [I-D.ietf-ace-key-groupcomm-oscore], this parameter specifies N, i.e., the maximum number of sets of stale OSCORE Sender IDs that the Group Manager stores for the group.

* 'exp', defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm].

* 'gid_reuse', encoding the CBOR simple value true (0xf5) if, upon rekeying the OSCORE group, the Group Manager can reassign the values of the OSCORE Group ID used as OSCORE ID Context, as per Section 3.2.1.1 of [I-D.ietf-core-oscore-groupcomm] and Section 11 of [I-D.ietf-ace-key-groupcomm-oscore]. Otherwise, this parameter encodes the CBOR simple value false (0xf4).

* 'app_groups', with value a list of names of application groups, encoded as a CBOR array. Each element of the array is a CBOR text string, specifying the name of an application group using the OSCORE group as security group (see Section 2.1 of [I-D.ietf-core-groupcomm-bis]).

* 'joining_uri', with value the URI of the group-membership resource for joining the newly created OSCORE group as per Section 6.2 of [I-D.ietf-ace-key-groupcomm-oscore], encoded as a CBOR text string.

* 'group_policies', defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], and consistent with the format and content defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

* 'as_uri', with value the URI of the Authorization Server associated with the Group Manager for the OSCORE group, encoded as a CBOR text string. Candidate group members will have to obtain an Access Token from that Authorization Server, before starting the joining process with the Group Manager to join the OSCORE group (see Sections 5 and 6 of [I-D.ietf-ace-key-groupcomm-oscore]).

5.2. Default Values

This section defines the default values that the Group Manager refers to for configuration and status parameters.

The Group Manager MAY choose different default values instead of those recommended in this section. A possible reason is to ensure that each of those are consistent with what the Group Manager supports, e.g., in terms of signature algorithm and format of authentication credentials used in the OSCORE group.

This ensures that the Group Manager is able to perform the operations defined in [I-D.ietf-ace-key-groupcomm-oscore], as to its interactions with joining nodes and current group members for an OSCORE group (see Section 14 of [I-D.ietf-ace-key-groupcomm-oscore]).

5.2.1. Configuration Parameters

For each of the configuration parameters listed below, the Group Manager refers to the following pre-configured default value, if none is specified by the Administrator.

* For 'group_mode', the Group Manager SHOULD use the CBOR simple value true (0xf5).

* If 'group_mode' has value true (0xf5), the Group Manager SHOULD use the default values defined in Section 14.2 of [I-D.ietf-ace-key-groupcomm-oscore] as follows: the value of 'sign_enc_alg' for the parameter 'gp_enc_alg' defined in this document; the value of 'sign_alg' and 'sign_params' for the corresponding parameters defined in this document.

  Editor's note: as per the text above, the referred version of [I-D.ietf-ace-key-groupcomm-oscore] still uses 'sign_enc_alg' as parameter name. The next version of [I-D.ietf-ace-key-groupcomm-oscore] will be updated in order to use 'gp_enc_alg' instead, as already done for this document and consistently with the naming used in the latest version of [I-D.ietf-core-oscore-groupcomm].

* If 'group_mode' has value true (0xf5), the Group Manager SHOULD use the CBOR simple value false (0xf4) for the parameter 'det_req'.

* If 'det_req' has value true (0xf5), the Group Manager SHOULD use SHA-256 (COSE algorithm encoding: -16) as default value for the parameter 'det_hash_alg'.

* For 'pairwise_mode', the Group Manager SHOULD use the CBOR simple value true (0xf5).

* If 'pairwise_mode' has value true (0xf5), the Group Manager SHOULD use the same default values defined in Section 14.3 of [I-D.ietf-ace-key-groupcomm-oscore] for the parameters 'alg', 'ecdh_alg', and 'ecdh_params'.

* For any other configuration parameter, the Group Manager SHOULD use the same default values defined in Section 14.1 of [I-D.ietf-ace-key-groupcomm-oscore].

5.2.2. Status Parameters

For each of the status parameters listed below, the Group Manager refers to the following pre-configured default value, if none is specified by the Administrator.

* For 'active', the Group Manager SHOULD use the CBOR simple value false (0xf4).

* For 'group_description', the Group Manager SHOULD use the CBOR simple value null (0xf6).

* For 'max_stale_sets', the Group Manager SHOULD use the CBOR unsigned integer with value 3, consistent with what is defined in Section 14.1 of [I-D.ietf-ace-key-groupcomm-oscore].

* For 'gid_reuse', the Group Manager SHOULD use the CBOR simple value false (0xf4).

* For 'app_groups', the Group Manager SHOULD use the empty CBOR array.

* For 'group_policies', the Group Manager SHOULD use the default values defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore].

6. Interactions with the Group Manager

This section describes the operations that are possible to perform on the group-collection resource and the group-configuration resources at the Group Manager.

For each operation, it is defined whether that operation is required or optional to support for the Group Manager and an Administrator. If the Group Manager supports an operation, then the Group Manager must be able to correctly handle authorized and valid requests sent by the Administrator to carry out that operation. If the Group Manager receives an authorized and valid request to perform an operation that it does not support, then the Group Manager MUST respond with a 5.01 (Not Implemented) response.

When checking the scope claim of a stored Access Token to verify that any of the requests defined in the following is authorized, the Group Manager only considers scope entries expressing permissions for administrative operations, namely "admin scope entries" as defined in Section 3. The alternative "user scope entries" defined in [I-D.ietf-ace-key-groupcomm-oscore] are not considered. That is, when handling any of the requests for administrative operations defined in the following, the Group Manager ignores possible "user scope entries" specified in the scope of a stored access token.

The Content-Format in messages containing a payload is set to application/ace-groupcomm+cbor, defined in Section 11.2 of [I-D.ietf-ace-key-groupcomm]. Furthermore, the entry labels defined in Section 7 of this document MUST be used when specifying the corresponding configuration and status parameters.

6.1. Retrieve the Full List of Group Configurations

This operation MUST be supported by the Group Manager and an Administrator.

The Administrator can send a GET request to the group-collection resource, in order to retrieve a list of the existing OSCORE groups at the Group Manager. This is returned as a list of links to the corresponding group-configuration resources.

The Group Manager MUST prepare the list L to include in the response as follows. For each group-configuration resource R:

1. The Group Manager considers the group name GROUPNAME of the OSCORE group associated with R.

2. The Group Manager retrieves the stored Access Token for the Administrator. Then, it checks whether GROUPNAME matches with the group name pattern specified in any scope entry of the 'scope' claim in the Access Token.

3. The link to the group-configuration resource R is added to the list L only in case of a positive match.

An example of message exchange is shown below.

```
=> 0.01 GET
   Uri-Path: manage

<= 2.05 Content
   Content-Format: 40 (application/link-format)

   Payload:

   <coap://[2001:db8::ab]/manage/gp1>;rt="core.osc.gconf",
   <coap://[2001:db8::ab]/manage/gp2>;rt="core.osc.gconf",
   <coap://[2001:db8::ab]/manage/gp3>;rt="core.osc.gconf"
```

6.2. Retrieve a List of Group Configurations by Filters

This operation MUST be supported by the Group Manager and MAY be supported by an Administrator.

The Administrator can send a FETCH request to the group-collection resource, in order to retrieve a list of the existing OSCORE groups that fully match a set of specified filter criteria. This is returned as a list of links to the corresponding group-configuration resources.

The filter criteria are specified in the request payload as a CBOR map, whose possible entries are specified in Section 5.1 and use the same abbreviations defined in Section 7.

Entry values are the ones admitted for the corresponding labels in the POST request for creating a group configuration (see Section 6.3), with the exception that the parameter 'group_name' (if present) can also be encoded as a tagged CBOR data item, specifying a group name pattern with the semantics signaled by the CBOR tag.

In such a case, the parameter 'group_name' expresses a group name pattern in the same way as a complex pattern Toid does in a scope entry (see Section 3). In particular, the filter criterion is satisfied by any group name that matches with the group name pattern specified by the parameter 'group_name' in the payload of the FETCH request.

The Group Manager MUST prepare the list L to include in the response as follows.

1. The Group Manager prepares a preliminary version of the list L, as specified in Section 6.1 for the processing of a GET request to the group-collection resource.

2. The Group Manager applies the filter criteria specified in the FETCH request to the list L from the previous step. The result is the list L to include in the response.

An example of message exchange is shown below.

```
=> 0.05 FETCH
   Uri-Path: manage
   Content-Format: CT_TBD (application/ace-groupcomm+cbor)

   Payload:

   {
       "group_mode" : true,
       "gp_enc_alg" : 10,
             "hkdf" : 5
   }

<= 2.05 Content
   Content-Format: 40 (application/link-format)

   Payload:

   <coap://[2001:db8::ab]/manage/gp1>;rt="core.osc.gconf",
   <coap://[2001:db8::ab]/manage/gp2>;rt="core.osc.gconf",
   <coap://[2001:db8::ab]/manage/gp3>;rt="core.osc.gconf"
```

The following, additional example considers a request payload that uses both configuration parameters and status parameters as filter criteria. In particular, the following is assumed.

* In the URI of every group-configuration resource, the path component is /manage/GROUPNAME, where GROUPNAME is the group name of the associated OSCORE group.

* The Group Manager hosts four group-configuration resources, for the OSCORE groups with names "gp4", "gp5", "gp6", and "foo". All the OSCORE groups use the AEAD Algorithm AES-CCM-16-64-128 (COSE algorithm encoding: 10). Also, all the OSCORE groups are currently active, except for "gp6".

```
=> 0.05 FETCH
   Uri-Path: manage
   Content-Format: CT_TBD (application/ace-groupcomm+cbor)

   Payload:

   {
       "gp_enc_alg" : 10,
       "group_name" : 21065("^gp[0-9]*$"),
           "active" : true
   }

<= 2.05 Content
   Content-Format: 40 (application/link-format)

   Payload:

   <coap://[2001:db8::ab]/manage/gp4>;rt="core.osc.gconf",
   <coap://[2001:db8::ab]/manage/gp5>;rt="core.osc.gconf"
```

6.3. Create a New Group Configuration

This operation MUST be supported by the Group Manager and an Administrator.

The Administrator can send a POST request to the group-collection resource, in order to create a new OSCORE group at the Group Manager. The request MUST specify the intended group name GROUPNAME, and MAY specify the intended group title together with pieces of information concerning the group configuration.

The request payload is a CBOR map, whose possible entries are specified in Section 5.1 and use the same abbreviations defined in Section 7. In particular:

* The payload MAY include any of the configuration parameters defined in Section 5.1.1.

* The payload MUST include the status parameter 'group_name' defined in Section 5.1.2 and specifying the intended group name encoded as a CBOR text string.

* The payload MAY include any of the status parameters 'active', 'group_description', 'max_stale_sets', 'exp', 'gid_reuse', 'app_groups', 'group_policies', and 'as_uri' defined in Section 5.1.2.

* The payload MUST NOT include any of the status parameters 'rt', 'ace_groupcomm_profile', and 'joining_uri' defined in Section 5.1.2.

Consistently with what is defined at step 4 of Section 4, the Group Manager MUST check whether the group name specified in the 'group_name' parameter matches with the group name pattern specified in any scope entry of the 'scope' claim in the stored Access Token for the Administrator. In case of a positive match, the Group Manager MUST check whether the permission set in the found scope entry specifies the permission "Create".

If the verification above fails (i.e., there are no matching scope entries specifying the "Create" permission), the Group Manager MUST reply with a 4.03 (Forbidden) error response.

If the group configuration to be created would include parameter values that prevent the Group Manager from performing the operations defined in [I-D.ietf-ace-key-groupcomm-oscore] (e.g., due to the Group Manager not supporting a format of authentication credentials), the Group Manager MUST respond with a 5.03 (Service Unavailable) response. The response MUST have Content-Format set to application/concise-problem-details+cbor [RFC9290] and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 12 ("Unsupported group configuration"), and the 'detail' field SHOULD be included in order to provide additional context.

Otherwise, if any of the following occurs, the Group Manager MUST respond with a 4.00 (Bad Request) response.

* Any of the received parameters is not recognized, or not valid, or not consistent with respect to other related parameters. In particular, the value of the status parameter 'exp' (if present) is not valid if the indicated expiration date is not in the future.

* The Group Manager does not trust the Authorization Server with URI specified in the 'as_uri' parameter, and has no alternative Authorization Server to consider for the OSCORE group to create.

After a successful processing of the POST request, the Group Manager performs the following actions.

If the 'group_name' parameter specifies the group name of an already existing OSCORE group, the Group Manager MUST try to determine an alternative name for the new OSCORE group to create.

In addition to that, the final decision about the name assigned to the new OSCORE group is always of the Group Manager, which may have more constraints than the Administrator can be aware of, possibly beyond the availability of suggested names. For example, the Group Manager may specifically want to use a randomized character string as the name of a newly created group.

If the Group Manager has selected a name GROUPNAME different from the name GROUPNAME* indicated in the parameter 'group_name' of the request, then the following conditions MUST hold.

* The chosen name GROUPNAME is available to assign; and

* With reference to the 'scope' claim in the stored Access Token for the Administrator, let us define PERM* as the union of the permission sets associated with the scope entries such that GROUPNAME* matches with the specified group name pattern. Also, let us define PERM as the union of the permission sets associated with the scope entries such that GROUPNAME matches with the specified group name pattern. Then, PERM and PERM* MUST be equal.

If the Group Manager does not manage to determine a group name for which both the above conditions hold, the Group Manager MUST respond with a 5.03 (Service Unavailable) response. The response MUST have Content-Format set to application/concise-problem-details+cbor [RFC9290] and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 11 ("Unable to determine a group name").

Otherwise, the Group Manager creates a new group-configuration resource, accessible to the Administrator at /manage/GROUPNAME, where GROUPNAME is the name of the OSCORE group as either indicated in the parameter 'group_name' of the request or uniquely assigned by the Group Manager. The group-collection resource is also accordingly updated.

The operation of creating the new group-configuration resource and accordingly updating the group-collection resource MUST be atomic.

The value of the status parameter 'rt' is set to "core.osc.gconf". The values of other parameters specified in the request are used as group configuration information for the newly created OSCORE group.

If the request specifies the parameter 'gid_reuse' encoding the CBOR simple value true (0xf5) and the Group Manager does not support the reassignment of OSCORE Group ID values (see Section 3.2.1.1 of [I-D.ietf-core-oscore-groupcomm] and Section 11 of

[I-D.ietf-ace-key-groupcomm-oscore]), then the Group Manager sets the value of the 'gid_reuse' status parameter in the group-configuration resource to the CBOR simple value false (0xf4).

For each parameter not specified in the request, the Group Manager refers to the default values specified in Section 5.2.

After that, the Group Manager creates a new group-membership resource accessible at /ace-group/GROUPNAME to nodes that want to join the OSCORE group, as specified in Section 6.1 of [I-D.ietf-ace-key-groupcomm-oscore]. Note that such group membership-resource comprises a number of sub-resources intended to current group members, as defined in Section 4.1 of [I-D.ietf-ace-key-groupcomm] and Section 8 of [I-D.ietf-ace-key-groupcomm-oscore].

From then on, the Group Manager will rely on the current group configuration to build the Join Response message defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore], when handling the joining of a new group member. Furthermore, the Group Manager generates the following pieces of information, and assigns them to the newly created OSCORE group:

* The OSCORE Master Secret.

* The OSCORE Master Salt (optionally).

* The Group ID, used as OSCORE ID Context, which MUST be unique within the set of OSCORE groups under the Group Manager.

Finally, the Group Manager replies to the Administrator with a 2.01 (Created) response. The Location-Path option MUST be included in the response, indicating the location of the just created group-configuration resource. The response MUST NOT include a Location-Query option.

The response payload specifies the parameters 'group_name', 'joining_uri', and 'as_uri', from the status properties of the newly created OSCORE group (see Section 5.1), as detailed below.

The response payload is a CBOR map, where entries use the same abbreviations defined in Section 7.

* 'group_name', with value the group name of the OSCORE group. This value can be different from the group name possibly specified by the Administrator in the POST request, and reflects the final choice of the Group Manager as 'group_name' status property for the OSCORE group. This parameter MUST be included.

* 'joining_uri', with value the URI of the group-membership resource for joining the newly created OSCORE group. This parameter MUST be included.

* 'as_uri', with value the URI of the Authorization Server associated with the Group Manager for the newly created OSCORE group. This parameter MUST be included. Its value can be different from the URI possibly specified by the Administrator in the POST request, and reflects the final choice of the Group Manager as 'as_uri' status property for the OSCORE group.

If the POST request specified the parameter 'gid_reuse' encoding the CBOR simple value true (0xf5) but the Group Manager has set the value of the 'gid_reuse' status parameter in the group-configuration resource to the CBOR simple value false (0xf4), then the response payload MUST include also the parameter 'gid_reuse' encoding the CBOR simple value false (0xf4).

If the POST request did not specify certain parameters and the Group Manager used default values different from the ones recommended in Section 5.2, then the response payload MUST also include those parameters, specifying the values chosen by the Group Manager for the current group configuration.

The Group Manager can register the link to the group-membership resource with URI specified in 'joining_uri' to a Resource Directory [RFC9176], as defined in Section 2 of [I-D.tiloca-core-oscore-discovery]. The Group Manager considers the current group configuration when specifying additional information for the link to register.

Alternatively, the Administrator can perform the registration in the Resource Directory on behalf of the Group Manager, acting as Commissioning Tool. The Administrator considers the following when specifying additional information for the link to register.

* The name of the OSCORE group MUST take the value specified in 'group_name' from the 2.01 (Created) response.

* The names of the application groups using the OSCORE group MUST take the values possibly specified by the elements of the 'app_groups' parameter in the POST request.

* If also registering a related link to the Authorization Server associated with the OSCORE group, the related link MUST have as link target the URI in 'as_uri' from the 2.01 (Created) response.

* As to every other information element describing the current group configuration, the following applies.

  - If a certain parameter was specified in the POST request, the Administrator MUST use either the value specified in the 2.01 (Created) response, if the Group Manager specified one, or the value specified in the POST request otherwise.

  - If a certain parameter was not specified in the POST request, the Administrator MUST use either the value specified in the 2.01 (Created) response, if the Group Manager specified one, or the corresponding default value recommended in Section 5.2.1 otherwise.

Note that, compared to the Group Manager, the Administrator is less likely to remain closely aligned with possible changes and updates that would require a prompt update to the registration in the Resource Directory. This applies especially to the address of the Group Manager, as well as the URI of the group-membership resource or of the Authorization Server associated with the Group Manager.

Therefore, it is RECOMMENDED that registrations of links to group-membership resources in the Resource Directory are made (and possibly updated) directly by the Group Manager, rather than by the Administrator.

An example of message exchange is shown below.

```
=> 0.02 POST
   Uri-Path: manage
   Content-Format: CT_TBD (application/ace-groupcomm+cbor)

   Payload:

   {
            "gp_enc_alg" : 10,
                 "hkdf" : 5,
       "pairwise_mode" : true,
              "active" : true,
          "group_name" : "gp4",
    "group_description" : "rooms 1 and 2",
          "app_groups" : ["room1", "room2"],
              "as_uri" : "coap://as.example.com/token"
   }

<= 2.01 Created
   Location-Path: manage
   Location-Path: gp4
   Content-Format: CT_TBD (application/ace-groupcomm+cbor)

   Payload:

   {
      "group_name" : "gp4",
    "joining_uri" : "coap://[2001:db8::ab]/ace-group/gp4/",
         "as_uri" : "coap://as.example.com/token"
   }
```

6.4. Retrieve a Group Configuration

This operation MUST be supported by the Group Manager and an Administrator.

The Administrator can send a GET request to the group-configuration resource manage/GROUPNAME associated with an OSCORE group with group name GROUPNAME, in order to retrieve the complete current configuration of that group.

Consistently with what is defined at step 4 of Section 4, the Group Manager MUST check whether GROUPNAME matches with the group name pattern specified in any scope entry of the 'scope' claim in the stored Access Token for the Administrator. In case of a positive match, the Group Manager MUST check whether the permission set in the found scope entry specifies the permission "Read".

If the verification above fails (i.e., there are no matching scope entries specifying the "Read" permission), the Group Manager MUST reply with a 4.03 (Forbidden) error response.

Otherwise, after a successful processing of the GET request, the Group Manager replies to the Administrator with a 2.05 (Content) response. The response has as payload the representation of the group configuration as specified in Section 5.1. The exact content of the payload reflects the current configuration of the OSCORE group. This includes both configuration properties and status properties.

The response payload is a CBOR map, whose possible entries are specified in Section 5.1 and use the same abbreviations defined in Section 7.

An example of message exchange is shown below.

```
  => 0.01 GET
     Uri-Path: manage
     Uri-Path: gp4

  <= 2.05 Content
     Content-Format: CT_TBD (application/ace-groupcomm+cbor)

     Payload:

     {
                         "hkdf" : 5,
                    "cred_fmt" : 33,
                  "group_mode" : true,
                  "gp_enc_alg" : 10,
                    "sign_alg" : -8,
                "sign_params" : [[1], [1, 6]],
             "pairwise_mode" : true,
                       "alg" : 10,
                  "ecdh_alg" : -27,
                "ecdh_params" : [[1], [1, 6]],
                    "det_req" : false,
                         "rt" : "core.osc.gconf",
                     "active" : true,
                 "group_name" : "gp4",
         "group_description" : "rooms 1 and 2",
       "ace_groupcomm_profile" : "coap_group_oscore_app",
              "max_stale_sets" : 3,
                         "exp" : 1360289224,
                  "gid_reuse" : false,
                 "app_groups" : ["room1", "room2"],
                "joining_uri" : "coap://[2001:db8::ab]/ace-group/gp4/",
                     "as_uri" : "coap://as.example.com/token"
     }
```

6.5. Retrieve Part of a Group Configuration by Filters

This operation MUST be supported by the Group Manager and MAY be supported by an Administrator.

The Administrator can send a FETCH request to the group-configuration resource manage/GROUPNAME associated with an OSCORE group with group name GROUPNAME, in order to retrieve part of the current configuration of that group.

The request payload is a CBOR map, which contains the following field:

* 'conf_filter', encoded as a CBOR array. Each element of the array specifies one requested configuration parameter or status parameter of the current group configuration (see Section 5.1), encoded with the corresponding CBOR abbreviation defined in Section 7.

The Group Manager MUST perform the same authorization checks defined for the processing of a GET request to a group-configuration resource in Section 6.4. That is, the Group Manager MUST verify that the Administrator has been granted a "Read" permission applicable to the targeted group-configuration resource.

After a successful processing of the FETCH request, the Group Manager replies to the Administrator with a 2.05 (Content) response. The response has as payload a partial representation of the group configuration (see Section 5.1). The exact content of the payload reflects the current configuration of the OSCORE group, and is limited to the configuration properties and status properties requested by the Administrator in the FETCH request.

The response payload includes the requested configuration parameters and status parameters, and is formatted as in the response payload of a GET request to a group-configuration resource (see Section 6.4). If the request payload specifies a parameter that is not included in the group configuration, then the response payload MUST NOT include a corresponding parameter.

An example of message exchange is shown below.

```
=> 0.05 FETCH
   Uri-Path: manage
   Uri-Path: gp4
   Content-Format: CT_TBD (application/ace-groupcomm+cbor)

   Payload:

   {
     "conf_filter" : ["gp_enc_alg",
                      "hkdf",
                      "pairwise_mode",
                      "active",
                      "group_description",
                      "app_groups"]
   }

<= 2.05 Content
   Content-Format: CT_TBD (application/ace-groupcomm+cbor)

   Payload:

   {
            "gp_enc_alg" : 10,
                  "hkdf" : 5,
         "pairwise_mode" : true,
                "active" : true,
     "group_description" : "rooms 1 and 2",
            "app_groups" : ["room1", "room2"]
   }
```

6.6. Overwrite a Group Configuration

This operation MAY be supported by the Group Manager and an Administrator.

The Administrator can send a PUT request to the group-configuration resource manage/GROUPNAME associated with an OSCORE group with group name GROUPNAME, in order to overwrite the current configuration of that group with a new one.

The payload of the request has the same format of the POST request defined in Section 6.3, with the exception that the configuration parameters 'group_mode' and 'pairwise_mode' as well as the status parameters 'group_name' and 'gid_reuse' MUST NOT be included.

The error handling for the PUT request is the same as for the POST request defined in Section 6.3, with the following difference in terms of authorization checks.

Consistently with what is defined at step 4 of Section 4, the Group Manager MUST check whether GROUPNAME matches with the group name pattern specified in any scope entry of the 'scope' claim in the stored Access Token for the Administrator. In case of a positive match, the Group Manager MUST check whether the permission set in the found scope entry specifies the permission "Write".

If the verification above fails (i.e., there are no matching scope entries specifying the "Write" permission), the Group Manager MUST reply with a 4.03 (Forbidden) error response.

If the group-configuration resource targeted by the PUT request does not currently exist, then the Group Manager MUST NOT create the resource and MUST reply with a 4.04 (Not Found) error response.

If the updated group configuration would include parameter values that prevent the Group Manager from performing the operations defined in [I-D.ietf-ace-key-groupcomm-oscore] (e.g., due to the Group Manager not supporting a format of authentication credentials), the Group Manager MUST respond with a 5.03 (Service Unavailable) response. The response MUST have Content-Format set to application/concise-problem-details+cbor [RFC9290] and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 12 ("Unsupported group configuration"), and the 'detail' field SHOULD be included in order to provide additional context.

If no error occurs and the PUT request is successfully processed, the Group Manager performs the following actions.

First, the Group Manager updates the group-configuration resource, consistently with the values indicated in the PUT request from the Administrator. When doing so, the configuration parameters 'group_mode' and 'pairwise_mode' as well as the status parameters 'group_name' and 'gid_reuse' MUST remain unchanged.

For each other parameter not specified in the PUT request, the Group Manager MUST use default values as specified in Section 5.2. Note that the default value recommended for the status parameter 'active' is the CBOR simple value false (0xf4). Therefore, if the Administrator intends to preserve the current status of the group as active, then the payload of the PUT request has to include the parameter 'active' specifying the CBOR simple value true (0xf5).

When updating the group-configuration resource, the corresponding group-membership resource is also accordingly updated. The overwriting of such two resources MUST be atomic.

If a new value N' is specified for the 'max_stale_sets' status parameter and N' is smaller than the current value N, the Group Manager preserves the (up to) N' most recent sets of stale OSCORE Sender IDs associated with the group, and deletes any possible older set (see Section 7.1 of [I-D.ietf-ace-key-groupcomm-oscore]).

From then on, the Group Manager relies on the latest updated configuration to build the Join Response message defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore], when handling the joining of a new group member. Similarly, the Group Manager relies on the new group configuration when building responses specifying (part of) the group configuration to a current group member. For instance, this applies when a group member retrieves from the Group Manager the updated group keying material (see Section 9.1 of [I-D.ietf-ace-key-groupcomm-oscore]) or the current group status (see Section 9.9 of [I-D.ietf-ace-key-groupcomm-oscore]).

Then, the Group Manager replies to the Administrator with a 2.04 (Changed) response. The payload of the response has the same format of the 2.01 (Created) response defined in Section 6.3.

If the PUT request did not specify certain parameters and the Group Manager used default values different from the ones recommended in Section 5.2, then the response payload MUST include also those parameters, specifying the values chosen by the Group Manager for the current group configuration.

If the link to the group-membership resource was registered in the Resource Directory [RFC9176], the Group Manager is responsible to refresh the registration, as defined in Section 3 of [I-D.tiloca-core-oscore-discovery].

Alternatively, the Administrator can update the registration in the Resource Directory on behalf of the Group Manager, acting as Commissioning Tool. The Administrator considers the following when specifying additional information for the link to update.

* The name of the OSCORE group MUST take the value specified in 'group_name' from the 2.04 (Changed) response.

* The names of the application groups using the OSCORE group MUST take the values possibly specified by the elements of the 'app_groups' parameter in the PUT request.

* If also registering a related link to the Authorization Server associated with the OSCORE group, the related link MUST have as link target the URI in 'as_uri' from the 2.04 (Changed) response.

* As to every other information element describing the current group configuration, the following applies.

  - If a certain parameter was specified in the PUT request, the Administrator MUST use either the value specified in the 2.04 (Changed) response, if the Group Manager specified one, or the value specified in the PUT request otherwise.

  - If a certain parameter was not specified in the PUT request, the Administrator MUST use either the value specified in the 2.04 (Changed) response, if the Group Manager specified one, or the corresponding default value recommended in Section 5.2.1 otherwise.

As discussed in Section 6.3, it is RECOMMENDED that registrations of links to group-membership resources in the Resource Directory are made (and possibly updated) directly by the Group Manager, rather than by the Administrator.

An example of message exchange is shown below.

```
=> 0.03 PUT
   Uri-Path: manage
   Uri-Path: gp4
   Content-Format: CT_TBD (application/ace-groupcomm+cbor)

   Payload:

   {
       "gp_enc_alg" : 11,
             "hkdf" : 5
   }

<= 2.04 Changed
   Content-Format: CT_TBD (application/ace-groupcomm+cbor)

   Payload:

   {
      "group_name" : "gp4",
    "joining_uri" : "coap://[2001:db8::ab]/ace-group/gp4/",
         "as_uri" : "coap://as.example.com/token"
   }
```

6.6.1. Effects on Joining Nodes

After having overwritten a group configuration, if the value of the status parameter 'active' is changed from true (0xf5) to false (0xf4), the Group Manager MUST stop admitting new members in the OSCORE group. In particular, until the status parameter 'active' is changed back to true (0xf5), the Group Manager MUST respond to a Join Request with a 5.03 (Service Unavailable) response, as defined in Section 6.2 of [I-D.ietf-ace-key-groupcomm-oscore].

If the value of the status parameter 'active' is changed from false (0xf4) to true (0xf5), the Group Manager resumes admitting new members in the OSCORE group, by processing their Join Requests (see Section 6.2 of [I-D.ietf-ace-key-groupcomm-oscore]).

6.6.2. Effects on the Group Members

After having overwritten a group configuration, the Group Manager informs the members of the OSCORE group, over the pairwise secure communication channels established when joining the group (see Section 6 of [I-D.ietf-ace-key-groupcomm-oscore]).

To this end, the Group Manager can individually target the 'control_uri' URI of each group member (see Section 4.3.1 of [I-D.ietf-ace-key-groupcomm]), if provided by the intended recipient upon joining the OSCORE group (see Section 6.1 of [I-D.ietf-ace-key-groupcomm-oscore]). Such messages sent by the Group Manager to each group member MUST have Content-Format set to application/ace-groupcomm+cbor, and MUST be formatted as the Join Response defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore], with the following differences.

* Only the parameters 'gkty', 'key', 'num', 'exp', and 'ace_groupcomm_profile' are present.

* The 'key' parameter includes only the following parameters, with values reflecting the new configuration of the OSCORE group.

  - 'hkdf' and 'cred_fmt'.

  - 'sign_enc_alg', 'sign_alg', and 'sign_params', only in case the configuration parameter 'group_mode' in the group-configuration resource has value true (0xf5), i.e., the OSCORE group uses the group mode of Group OSCORE.

    Editor's note: as per the text above, the referred version of [I-D.ietf-ace-key-groupcomm-oscore] still uses 'sign_enc_alg' as parameter name. The next version of

[I-D.ietf-ace-key-groupcomm-oscore] will be updated in order to use 'gp_enc_alg' instead, as already done for this document and consistently with the naming used in the latest version of [I-D.ietf-core-oscore-groupcomm].

- 'alg', 'ecdh_alg', and 'ecdh_params', only in case the configuration parameter 'pairwise_mode' in the group-configuration resource has value true (0xf5), i.e., the OSCORE group uses the pairwise mode of Group OSCORE.

- 'det_hash_alg' defined in Section 4 of [I-D.amsuess-core-cachable-oscore], only in case the configuration parameter 'det_req' is present with value true (0xf5), and specifying the Hash Algorithm used in the OSCORE group when producing deterministic requests, as defined in [I-D.amsuess-core-cachable-oscore].

Alternatively, group members can obtain the information above by accessing the group-membership resource associated with the OSCORE group (see Section 9.1 of [I-D.ietf-ace-key-groupcomm-oscore]), optionally by subscribing for updates to such a resource, e.g., by using CoAP Observe [RFC7641].

When receiving such information, each group member uses it to update the corresponding parameters in the Group OSCORE Security Context of the group in question (see Section 2 of [I-D.ietf-core-oscore-groupcomm]. If any of 'sign_enc_alg', 'sign_alg', 'alg', and 'ecdh_alg' has as value the CBOR simple value null (0xf6), then the corresponding parameter in the Group OSCORE Security Context becomes unset if it is not already. According to the new parameter values, each group member derives new Sender/Recipient Keys, a new Common IV, and new Pairwise Keys. When doing so, a group member MUST NOT reset the Sender Sequence Number in its Sender Context or reset the Replay Window in its Recipient Contexts.

Editor's note: as per the text above, the referred version of [I-D.ietf-ace-key-groupcomm-oscore] still uses 'sign_enc_alg' as parameter name. The next version of [I-D.ietf-ace-key-groupcomm-oscore] will be updated in order to use 'gp_enc_alg' instead, as already done for this document and consistently with the naming used in the latest version of [I-D.ietf-core-oscore-groupcomm].

The following holds when the value of specific parameters is updated.

* If the value of the status parameter 'active' is changed from true (0xf5) to false (0xf4):

- The Group Manager MUST stop accepting requests for new individual keying material from current group members (see Section 9.2 of [I-D.ietf-ace-key-groupcomm-oscore]), until the status parameter 'active' is changed back to true (0xf5). Until then, the Group Manager MUST respond to a Key Renewal Request with a 5.03 (Service Unavailable) response, as defined in Section 9.2 of [I-D.ietf-ace-key-groupcomm-oscore].

- The Group Manager MUST stop accepting updated authentication credentials uploaded by current group members (see Section 9.4 of [I-D.ietf-ace-key-groupcomm-oscore]), until the status parameter 'active' is changed back to true (0xf5). Until then, the Group Manager MUST respond to an Authentication Credential Update Request with a 5.03 (Service Unavailable) response, as defined in Section 9.4 of [I-D.ietf-ace-key-groupcomm-oscore].

* Every group member, upon learning that the OSCORE group has been deactivated (i.e., 'active' has value false (0xf4)), SHOULD stop communicating in the group.

  Every group member, upon learning that the OSCORE group has been reactivated (i.e., 'active' has value true (0xf5) again), can resume communicating in the group.

* If the value of 'gp_enc_alg' and/or 'alg' is changed, the Group Manager determines the new maximum size NEW_MAX_SIZE that can be used for the OSCORE Sender IDs of the group members, based on the size of the AEAD nonce of such algorithms (see Section 2.2 of [I-D.ietf-core-oscore-groupcomm]). In case NEW_MAX_SIZE is strictly smaller than the old, maximum size of the OSCORE Sender IDs used in the OSCORE group, the Group Manager MUST proceed as follows.

  - The Group Manager checks if any of the current group members has an OSCORE Sender ID whose size is strictly larger than NEW_MAX_SIZE.

  - If any of such group members is found, the Group Manager MUST evict them from the OSCORE group. That is, the Group Manager MUST terminate their membership and MUST rekey the group in such a way that the new keying material is not provided to those evicted members. This also includes adding their relinquished Sender IDs to the most recent set of stale Sender IDs for the OSCORE group (see Section 7.1 of [I-D.ietf-ace-key-groupcomm-oscore]), before rekeying the group. Such evicted group members can rejoin the OSCORE group, thus obtaining the new group keying material together with a new, valid OSCORE Sender ID.

* Every group member, upon receiving updated values for 'hkdf', 'sign_enc_alg', and 'alg', MUST either:

  - Leave the OSCORE group (see Section 9.11 of [I-D.ietf-ace-key-groupcomm-oscore]), e.g., if not supporting the indicated new algorithms; or

  - Remain in the OSCORE group and use the Group OSCORE Security Context after having updated it as defined above.

  Editor's note: as per the text above, the referred version of [I-D.ietf-ace-key-groupcomm-oscore] still uses 'sign_enc_alg' as parameter name. The next version of [I-D.ietf-ace-key-groupcomm-oscore] will be updated in order to use 'gp_enc_alg' instead, as already done for this document and consistently with the naming used in the latest version of [I-D.ietf-core-oscore-groupcomm].

* Every group member, upon receiving updated values for 'cred_fmt', 'sign_alg', 'sign_params', 'ecdh_alg', and 'ecdh_params' MUST either:

  - Leave the OSCORE group, e.g., if not supporting the indicated new format, algorithms, parameters, and encoding; or

  - Leave the OSCORE group and rejoin it (see Section 6 of [I-D.ietf-ace-key-groupcomm-oscore]). When rejoining the group, an authentication credential in the indicated format used in the OSCORE group MUST be provided to the Group Manager. The authentication credential as well as the included public key MUST be compatible with the indicated algorithms and parameters.

  - Remain in the OSCORE group and use the Group OSCORE Security Context after having updated it as defined above, and, if required, perform the following actions.

    o Provide the Group Manager with a new authentication credential to use in the OSCORE group (see Section 9.4 of [I-D.ietf-ace-key-groupcomm-oscore]). The new authentication credential MUST be in the indicated format used in the OSCORE group. The new authentication credential as well as the included public key MUST be compatible with the indicated algorithms and parameters.

      Consistently, the group member has to retrieve the new authentication credentials of other group members as they are uploaded to the Group Manager (see Section 9.3 of

[I-D.ietf-ace-key-groupcomm-oscore]). In order to ensure the retrieval of latest authentication credentials that are consistent with the new group configuration, it is preferable that the group member retrieves such authentication credentials after a pre-configured time interval has elapsed since uploading its own authentication credential. Later on, the group member will need to retrieve other group members' authentication credentials that it is still missing and that it needs for processing messages exchanged in the OSCORE group.

- o Retrieve from the Group Manager the Group Manager's new authentication credential (see Section 9.5 of [I-D.ietf-ace-key-groupcomm-oscore]). The new Group Manager's authentication credential is in the indicated format used in the OSCORE group. The new authentication credential as well as the included public key are compatible with the indicated algorithms and parameters.

6.7. Selective Update of a Group Configuration

This operation MAY be supported by the Group Manager and an Administrator.

The Administrator can send a PATCH/iPATCH request [RFC8132] to the group-configuration resource manage/GROUPNAME associated with an OSCORE group with group name GROUPNAME, in order to update the value of only part of the group configuration.

The request payload has the same format of the PUT request defined in Section 6.6, with the difference that it MAY also specify names of application groups to be removed from or added to the 'app_groups' status parameter. The names of such application groups are provided as defined below.

The CBOR map in the request payload includes the field 'app_groups_diff', whose CBOR abbreviation is defined in Section 7. This field is encoded as a CBOR array including the following two elements.

* The first element is a CBOR array, namely 'app_groups_del'. Each of its elements is a CBOR text string, with value the name of an application group to remove from the 'app_groups' status parameter.

* The second element is a CBOR array, namely 'app_groups_add'. Each of its elements is a CBOR text string, with value the name of an application group to add to the 'app_groups' status parameter.

The CDDL definition [RFC8610] of the CBOR array 'app_groups_diff' formatted as in the request from the Administrator is provided below.

```
app-group-name = tstr
name-patch = [* app-group-name]
app_groups_diff = [app_groups_del: name-patch,
                   app_groups_add: name-patch]
```

Figure 2: CDDL definition of the 'app_groups_diff' field

The Group Manager MUST respond with a 4.00 (Bad Request) response in case: both the inner CBOR arrays 'app_groups_del' and 'app_groups_add' are empty; or the CBOR map in the request payload includes both the 'app_groups' field and the 'app_groups_diff' field.

The error handling for the PATCH/iPATCH request is the same as for the PUT request defined in Section 6.6, with the following additions.

* The set of group configuration parameters to update MUST NOT be empty. That is, the Group Manager MUST respond with a 4.00 (Bad Request) response, if the request payload includes an empty CBOR map.

* If the Request-URI does not point to an existing group-configuration resource, the Group Manager MUST NOT create a new resource, and MUST respond with a 4.04 (Not Found) response.

* When applying the specified updated values would yield an inconsistent group configuration, the Group Manager MUST respond with a 4.09 (Conflict) response.

  The response MAY include the current representation of the group configuration resource, like when responding to a GET request as defined in Section 6.4. Otherwise, the response SHOULD include a diagnostic payload with additional information for the Administrator to recognize the source of the conflict.

* When the request uses specifically the iPATCH method, the Group Manager MUST respond with a 4.00 (Bad Request) response, in case the CBOR map includes the parameter 'app_groups_diff'.

Furthermore, the Group Manager MUST perform the same authorization checks defined for the processing of a PUT request to a group-configuration resource in Section 6.6. That is, the Group Manager MUST verify that the Administrator has been granted a "Write" permission applicable to the targeted group-configuration resource.

If the updated group configuration would include parameter values that prevent the Group Manager from performing the operations defined in [I-D.ietf-ace-key-groupcomm-oscore] (e.g., due to the Group Manager not supporting a format of authentication credentials), the Group Manager MUST respond with a 5.03 (Service Unavailable) response. The response MUST have Content-Format set to application/concise-problem-details+cbor [RFC9290] and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 12 ("Unsupported group configuration"), and the 'detail' field SHOULD be included in order to provide additional context.

If no error occurs and the PATCH/iPATCH request is successfully processed, the Group Manager performs the following actions.

First, the Group Manager updates the group-configuration resource, consistently with the values indicated in the PATCH/iPATCH request from the Administrator. The corresponding group-membership resource is also accordingly updated.

The operation of updating the group-configuration resource and accordingly updating the group-membership resource MUST be atomic.

Unlike for the PUT request defined in Section 6.6, the Group Manager does not alter the value of configuration parameters and status parameters for which updated values are not specified in the request payload. In particular, the Group Manager does not assign possible default values to those parameters.

Special processing occurs when updating the 'app_groups' status parameter by difference, as defined below. The Administrator should not expect the Group Manager to add or delete names of application group names according to any particular order.

* If the name of an application group to add (delete) is specified multiple times, the Group Manager considers it only once for addition to (deletion from) the 'app_groups' status parameter.

* If the name of an application group to delete is not present in the 'app_groups' status parameter before any change is applied, the Group Manager ignores that name.

* If the name of an application group to add is already present in the 'app_groups' status parameter before any change is applied, the Group Manager ignores that name.

* The Group Manager deletes from the 'app_groups' status parameter the names of the application groups specified in the inner 'app_groups_del' CBOR array of the 'app_groups_diff' field.

* The Group Manager adds to the 'app_groups' status parameter the names of the application groups specified in the inner 'app_groups_add' CBOR array of the 'app_groups_diff' field.

After having updated the group-configuration resource, from then on the Group Manager relies on the new group configuration to build the Join Response message defined in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore], when handling the joining of a new group member. Similarly, the Group Manager relies on the new group configuration when building responses specifying (part of) the group configuration to a current group member. For instance, this applies when a group member retrieves from the Group Manager the updated group keying material (see Section 9.1 of [I-D.ietf-ace-key-groupcomm-oscore]) or the current group status (see Section 9.9 of [I-D.ietf-ace-key-groupcomm-oscore]).

Finally, the Group Manager replies to the Administrator with a 2.04 (Changed) response. The payload of the response has the same format of the 2.01 (Created) response defined in Section 6.3.

The same considerations as for the PUT request defined in Section 6.6 hold also in this case, with respect to refreshing a possible registration of the link to the group-membership resource in the Resource Directory [RFC9176].

An example of message exchange is shown below.

```
   => 0.06 PATCH
      Uri-Path: manage
      Uri-Path: gp4
      Content-Format: CT_TBD (application/ace-groupcomm+cbor)

      Payload:

      {
             "gp_enc_alg" : 10,
        "app_groups_diff" : [["room1"],
                             ["room3", "room4"]]
      }

   <= 2.04 Changed
      Content-Format: CT_TBD (application/ace-groupcomm+cbor)

      Payload:

      {
         "group_name" : "gp4",
        "joining_uri" : "coap://[2001:db8::ab]/ace-group/gp4/",
             "as_uri" : "coap://as.example.com/token"
      }
```

6.7.1. Effects on Joining Nodes

After having selectively updated part of a group configuration, the effects on candidate joining nodes are the same as defined in Section 6.6.1 for the case of group configuration overwriting.

6.7.2. Effects on the Group Members

After having selectively updated part of a group configuration, the effects on the current group members are the same as defined in Section 6.6.2 for the case of group configuration overwriting.

6.8. Delete a Group Configuration

This operation MUST be supported by the Group Manager and an Administrator.

The Administrator can send a DELETE request to the group-configuration resource manage/GROUPNAME associated with an OSCORE group with group name GROUPNAME, in order to delete that OSCORE group.

Consistently with what is defined at step 4 of Section 4, the Group Manager MUST check whether GROUPNAME matches with the group name pattern specified in any scope entry of the 'scope' claim in the stored Access Token for the Administrator. In case of a positive match, the Group Manager MUST check whether the permission set in the found scope entry specifies the permission "Delete".

If the verification above fails (i.e., there are no matching scope entries specifying the "Delete" permission), the Group Manager MUST reply with a 4.03 (Forbidden) error response.

Otherwise, the Group Manager continues processing the request, which would be successful only on an inactive OSCORE group. That is, the DELETE request actually yields a successful deletion of the OSCORE group only if the corresponding status parameter 'active' has current value false (0xf4). The Administrator can ensure that, by first performing an update of the group-configuration resource associated with the OSCORE group (see Section 6.6), and setting the corresponding status parameter 'active' to false (0xf4).

If, upon receiving the DELETE request, the current value of the status parameter 'active' is true (0xf5), the Group Manager MUST respond with a 4.09 (Conflict) response. The response MUST have Content-Format set to application/concise-problem-details+cbor [RFC9290] and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 10 ("Group currently active").

After a successful processing of the DELETE request, the Group Manager performs the following actions.

First, the Group Manager deletes the OSCORE group, deallocates both the group-configuration resource as well as the group-membership resource associated with that group, and accordingly updates the group-collection resource.

The operation of deleting the group-configuration resource and the corresponding group-membership resource, as well as of accordingly updating the group-collection resource MUST be atomic.

Then, the Group Manager replies to the Administrator with a 2.02 (Deleted) response.

An example of message exchange is shown below.

```
=> 0.04 DELETE
   Uri-Path: manage
   Uri-Path: gp4

<= 2.02 Deleted
```

6.8.1. Effects on the Group Members

After having deleted an OSCORE group, the Group Manager can inform the group members by means of the following two methods. When contacting a group member, the Group Manager uses the pairwise secure communication association established with that member during its joining process (see Section 6 of [I-D.ietf-ace-key-groupcomm-oscore]).

* The Group Manager sends an individual request message to each group member, targeting the respective resource used to perform the group rekeying process (see Section 11.1 of [I-D.ietf-ace-key-groupcomm-oscore]). The Group Manager uses the same format of the Join Response message in Section 6.3 of [I-D.ietf-ace-key-groupcomm-oscore], where only the parameters 'gkty', 'key', and 'ace_groupcomm_profile' are present, and the 'key' parameter is the empty CBOR map.

* A group member may subscribe for updates to the group-membership resource associated with the OSCORE group. In particular, if this relies on CoAP Observe [RFC7641], a group member would receive a 4.04 (Not Found) notification response from the Group Manager, since the group-configuration resource has been deallocated upon deleting the OSCORE group (see Section 6.1 of [I-D.ietf-ace-key-groupcomm]). The response MUST have Content-Format set to application/concise-problem-details+cbor [RFC9290] and is formatted as defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]. Within the Custom Problem Detail entry 'ace-groupcomm-error', the value of the 'error-id' field MUST be set to 5 ("Group deleted").

When being informed about the deletion of the OSCORE group, a group member deletes the OSCORE Security Context that it stores as associated with that group, and possibly deallocates any dedicated control resource intended for the Group Manager that it has for that group.

7. ACE Groupcomm Parameters

In addition to what is defined in Section 8 of [I-D.ietf-ace-key-groupcomm], this document defines additional parameters used in the messages exchanged between the Administrator and the Group Manager (see Section 6). The table below summarizes them and specifies the CBOR key to use instead of the full descriptive name.

Note that the media type application/ace-groupcomm+cbor MUST be used when these parameters are transported in the respective message fields.

Name	CBOR Key	CBOR Type	Reference
hkdf	-1	tstr / int	[RFC-XXXX]
cred_fmt	-2	int	[RFC-XXXX]
group_mode	-3	simple value	[RFC-XXXX]
gp_enc_alg	-4	tstr / int / simple value	[RFC-XXXX]
sign_alg	-5	tstr / int / simple value	[RFC-XXXX]
sign_params	-6	array / simple value	[RFC-XXXX]
pairwise_mode	-7	simple value	[RFC-XXXX]
alg	-8	tstr / int / simple value	[RFC-XXXX]
ecdh_alg	-9	tstr / int / simple value	[RFC-XXXX]
ecdh_params	-10	array / simple value	[RFC-XXXX]
det_req	-25	simple value	[RFC-XXXX]
det_hash_alg	-26	tstr / int	[RFC-XXXX]
rt	-11	tstr	[RFC-XXXX]

active	-12	simple value	[RFC-XXXX]
group_name	-13	tstr / #6.<uint>(any)	[RFC-XXXX]
group_description	-14	tstr / simple value	[RFC-XXXX]
max_stale_sets	-15	uint	[RFC-XXXX]
gid_reuse	-16	simple value	[RFC-XXXX]
app_groups	-17	array	[RFC-XXXX]
joining_uri	-18	tstr	[RFC-XXXX]
as_uri	-19	tstr	[RFC-XXXX]
conf_filter	-27	array	[RFC-XXXX]
app_groups_diff	-28	array	[RFC-XXXX]

Table 2: ACE Groupcomm Parameters

The following holds for the Group Manager.

* It MUST support the parameters 'ace_groupcomm_profile', 'exp', and 'group_policies', which are defined in Section 8 of [I-D.ietf-ace-key-groupcomm].

  This is consistent with what is defined in Section 8 of [I-D.ietf-ace-key-groupcomm] for the Key Distribution Center, of which the Group Manager defined in [I-D.ietf-ace-key-groupcomm-oscore] is a specific instance.

* It MUST support all the parameters listed in Table 2, with the exception of the 'app_groups_diff' parameter, which MUST be supported only if the Group Manager supports the selective update of a group configuration (see Section 6.7).

The following holds for an Administrator.

* It MUST support the parameters 'ace_groupcomm_profile', 'exp', and 'group_policies', which are defined in Section 8 of [I-D.ietf-ace-key-groupcomm].

* It MUST support all the parameters listed in Table 2, with the following exceptions.

  - 'conf_filter', which MUST be supported only if the Administrator supports the partial retrieval of a group configuration by filters (see Section 6.5).

  - 'app_groups_diff' parameter, which MUST be supported only if the Administrator supports the selective update of a group configuration (see Section 6.7).

8. ACE Groupcomm Error Identifiers

In addition to what is defined in Section 9 of [I-D.ietf-ace-key-groupcomm], this document defines new values that the Group Manager can use as error identifiers. These are used in error responses with Content-Format application/concise-problem-details+cbor [RFC9290], as values of the 'error-id' field within the Custom Problem Detail entry 'ace-groupcomm-error' (see Section 4.1.2 of [I-D.ietf-ace-key-groupcomm]).

Value	Description	Reference
10	Group currently active	[RFC-XXXX]
11	Unable to determine a group name	[RFC-XXXX]
12	Unsupported group configuration	[RFC-XXXX]

Table 3: ACE Groupcomm Error Identifiers

If the Administrator supports the problem-details format [RFC9290] and the Custom Problem Detail entry 'ace-groupcomm-error' defined in Section 4.1.2 of [I-D.ietf-ace-key-groupcomm], and is able to understand the error specified in the 'error-id' field therein, then the Administrator may use that information to determine what actions to take next. If the Concise Problem Details data item specified in the error response includes the 'detail' entry and the Administrator supports it, such an entry may provide additional context. In particular, the following guidelines apply.

* In case of error 10, the Administrator should stop sending the DELETE request to the Group Manager (see Section 6.8), until the group becomes inactive. As per this document, this error is relevant only for the Administrator, if it tries to delete a group without having set its status to inactive first (see Section 6.8).

In such a case, the Administrator should take the expected course of actions, and set the group status to inactive first (see Section 6.6 and Section 6.7), before sending a new request of group deletion to the Group Manager.

* In case of error 11, the Administrator has the following options.

  - The Administrator simply tries again later on. The new POST request to the group-collection resource specifies the same group name originally suggested in the previous request that triggered the error response (see Section 6.3). This option fundamentally relies on the Group Manager making the group name available again before the Administrator sends the new POST request. Hence, this option is not viable if it is unacceptable for the Administrator to considerably or indefinitely postpone the creation of the new group.

  - The Administrator sends a new POST request to the group-collection resource right away, specifying a different group name than the one suggested in the previous request that triggered the error response.

    In order to not inadvertently suggest a new group name that is already assigned to an existing OSCORE group, the Administrator can first retrieve the list of existing groups from the Group Manager, as defined in Section 6.1. In the payload of the response from the Group Manager, each specified link indicates the name of an existing OSCORE group as the last segment of its url-path.

    The Administrator should choose the new group name GROUPNAME to suggest, in such a way that it does not renounce permissions that were granted per the old group name GROUPNAME*. This is the case if the following holds.

    With reference to the 'scope' claim of the Administrator's Access Token, let us define PERM* as the union of the permission sets associated with the scope entries such that GROUPNAME* matches with the specified group name pattern. Also, let us define PERM as the union of the permission sets associated with the scope entries such that GROUPNAME matches with the specified group name pattern. Then, PERM specifies no fewer permissions than PERM*.

  - The Administrator requests a new Access Token to the Authorization Server, in order to update its access rights, and have a new granted scope whose scope entries specify more and/or different group name patterns than the old Access Token.

After uploading the new Access Token to the Group Manager, the Administrator can send a new POST request to the group-collection resource. When doing so, the Administrator suggests a new group name to the Group Manager, according to the same criteria discussed for the previous option.

* In case of error 12, the Administrator has the following options.

  - If the Administrator has attempted to create a new group configuration (see Section 6.3), the Administrator can take into account what the Group Manager specifies in the 'detail' entry of the Concise Problem Details data item [RFC9290] specified in the error response, and send a new request to the Group Manager for accordingly creating the group configuration.

    This requires that the Administrator finds acceptable to create a group configuration different from the originally intended one.

  - If the Administrator has attempted to overwrite (see Section 6.6) or selectively update (see Section 6.7) an existing group configuration, the Administrator can take into account what the Group Manager specifies in the 'detail' entry of the Concise Problem Details data item [RFC9290] specified in the error response, and send a new request to the Group Manager for accordingly overwriting or selectively updating the group configuration.

    This requires that the Administrator finds acceptable to overwrite or update the current group configuration differently than how it was originally intended. If this is not attainable, the Administrator may decide to not take further actions and keep the current group configuration as is, or instead to delete the group configuration altogether (see Section 6.8).

9. Security Considerations

Security considerations are inherited from the ACE framework for Authentication and Authorization [RFC9200], and from the specific transport profile of ACE used between the Administrator and the Group Manager, such as [RFC9202] and [RFC9203].

The same security considerations from [I-D.ietf-ace-key-groupcomm] and [I-D.ietf-ace-key-groupcomm-oscore] also apply, with particular reference to the process of rekeying OSCORE groups.

Further security considerations are compiled below.

9.1. Change of Group Configuration

With respect to changing group configurations, the following security considerations hold.

* A change of the current group configuration (see Section 6.6 and Section 6.7) might result in generating and distributing new group keying material, consistently with the newly enforced algorithms and related parameters. In such a case, the Group Manager can perform a group rekeying as per Section 11 of [I-D.ietf-ace-key-groupcomm-oscore], or provide the new group keying material together with the new group configuration as per Section 6.6 and Section 6.7 of this document.

  After gaining knowledge of the new group configuration, current group members may also leave the OSCORE group and rejoin it, hence obtaining the new group configuration parameters and the up-to-date group keying material. When this happens, the Group Manager SHOULD NOT repeatedly rekey the group upon the re-join of every current group member, each of which is identifiable by means of the secure association that it has with the Group Manager.

  Shortly following an update of group configuration, the Group Manager SHOULD prioritize the re-join of such current group members before processing Join Requests from new group members.

* Following the enforcement of a new group configuration, a group member might not deem it conducive to a sufficient security level (e.g., in terms of security algorithms and related parameters). In such a case, it is RECOMMENDED that the group member leaves the group.

* A change of the current group configuration, possibly also requiring a group rekeying, might result in temporarily preventing communications among some group members altogether, until they have aligned themselves to the new group configuration. This is especially the case for a change of group configuration affecting the security algorithms and related parameters used in the group.

  Furthermore, a change of group configuration might interfere with ongoing, extended exchanges between group members, especially Block-Wise transfers [RFC7959][RFC9177] and the transmission of Observe notifications for ongoing Observations [RFC7641].

  A group configuration (possibly together with the group keying material) may have been updated while a Block-Wise transfer is ongoing between two group members. This will result in blocks being resent, if the block sender and recipient are not yet both

aligned with the new group configuration (and group keying material), in which case the block recipient would reply with an error message.

After a change of group configuration, a group member MUST terminate an ongoing Observation if the new group configuration would not have allowed to compute exactly the Observe request associated with the ongoing Observation. This occurs, for example, when the new group configuration specifies a signature algorithm different from the one used in the group when the Observe request was protected.

9.2. Group Manager

In addition to what is discussed in Section 10.1 of [I-D.ietf-ace-key-groupcomm], a compromised Group Manager would allow an adversary to also monitor the group configurations specified by an Administrator, or to enforce group configurations different from the specified ones, which can result in communications in the OSCORE groups not attaining the originally intended security level.

Although this is undesirable, it is not worse than the control that the adversary would gain on the group keying material through the compromised Group Manager (see Section 10.1 of [I-D.ietf-ace-key-groupcomm]).

Unlike what is defined in Section 10.2 of [I-D.ietf-ace-key-groupcomm] with respect to renewing the group keying material, the Group Manager does not have to change the group configurations of the OSCORE groups it is responsible for, after having experienced a reboot.

9.3. Administrators

If multiple Administrators are responsible for the same OSCORE group, they are expected to be aware of each other and of their shared responsibility, as well as to be aligned on what is in the best interest of the OSCORE group and its secure operation. It is out of the scope of this document to define how different Administrators are appointed as responsible for an OSCORE group, and how they achieve and maintain such an alignment with each other.

A compromised Administrator may collude with unauthorized parties. Within the extent of the granted access rights, the compromised Administrator may leak group configurations, change them in such a way that communications in the OSCORE groups do not attain the originally intended security level, or delete OSCORE groups altogether thus impeding their secure operation.

When an Administrator is found compromised, the pertaining Access Tokens MUST be revoked by the Authorization Server. A possible way for the Authorization Server to notify the affected Group Managers about such revoked Access Tokens is defined in [I-D.ietf-ace-revoked-token-notification].

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. ACE Groupcomm Parameters

IANA is asked to register the following entries in the "ACE Groupcomm Parameters" registry defined in Section 11.7 of [I-D.ietf-ace-key-groupcomm].

Name: hkdf
CBOR Key: -1
CBOR Type: tstr / int
Reference: [RFC-XXXX]

Name: cred_fmt
CBOR Key: -2
CBOR Type: int
Reference: [RFC-XXXX]

Name: group_mode
CBOR Key: -3
CBOR Type: simple value
Reference: [RFC-XXXX]

Name: gp_enc_alg
CBOR Key: -4
CBOR Type: tstr / int / simple value
Reference: [RFC-XXXX]

Name: sign_alg
CBOR Key: -5
CBOR Type: tstr / int / simple value
Reference: [RFC-XXXX]

Name: sign_params
CBOR Key: -6
CBOR Type: array / simple value
Reference: [RFC-XXXX]

Name: pairwise_mode
CBOR Key: -7
CBOR Type: simple value
Reference: [RFC-XXXX]

Name: alg
CBOR Key: -8
CBOR Type: tstr / int / simple value
Reference: [RFC-XXXX]

Name: ecdh_alg
CBOR Key: -9
CBOR Type: tstr / int / simple value
Reference: [RFC-XXXX]

Name: ecdh_params
CBOR Key: -10
CBOR Type: array / simple value
Reference: [RFC-XXXX]

Name: det_req
CBOR Key: -25
CBOR Type: simple value
Reference: [RFC-XXXX]

Name: det_hash_alg
CBOR Key: -26
CBOR Type: tstr / int
Reference: [RFC-XXXX]

Name: rt
CBOR Key: -11
CBOR Type: tstr
Reference: [RFC-XXXX]

Name: active
CBOR Key: -12
CBOR Type: simple value
Reference: [RFC-XXXX]

Name: group_name
CBOR Key: -13
CBOR Type: tstr / #6.<uint>(any)
Reference: [RFC-XXXX]

Name: group_description
CBOR Key: -14
CBOR Type: tstr / simple value

Reference: [RFC-XXXX]

Name: max_stale_sets
CBOR Key: -15
CBOR Type: uint
Reference: [RFC-XXXX]

Name: gid_reuse
CBOR Key: -16
CBOR Type: simple value
Reference: [RFC-XXXX]

Name: app_groups
CBOR Key: -17
CBOR Type: array
Reference: [RFC-XXXX]

Name: joining_uri
CBOR Key: -18
CBOR Type: tstr
Reference: [RFC-XXXX]

Name: as_uri
CBOR Key: -19
CBOR Type: tstr
Reference: [RFC-XXXX]

Name: conf_filter
CBOR Key: -27
CBOR Type: array
Reference: [RFC-XXXX]

Name: app_groups_diff
CBOR Key: -28
CBOR Type: array
Reference: [RFC-XXXX]

10.2. ACE Groupcomm Errors

IANA is asked to register the following entry in the "ACE Groupcomm Errors" registry defined in Section 11.12 of [I-D.ietf-ace-key-groupcomm].

Value: 10
Description: Group currently active
Reference: [RFC-XXXX]

Value: 11
Description: Unable to determine a group name
Reference: [RFC-XXXX]

Value: 12
Description: Unsupported group configuration
Reference: [RFC-XXXX]

10.3. Resource Types

IANA is asked to enter the following values in the "Resource Type (rt=) Link Target Attribute Values" registry within the "Constrained Restful Environments (CoRE) Parameters" registry group.

Value: core.osc.gcoll
Description: Group-collection resource of an OSCORE Group Manager
Reference: [RFC-XXXX]

Value: core.osc.gconf
Description: Group-configuration resource of an OSCORE Group Manager
Reference: [RFC-XXXX]

10.4. Group OSCORE Admin Permissions

This document establishes the IANA "Group OSCORE Admin Permissions" registry. The registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 10.5.

This registry includes the possible permissions that Administrators can have to perform operations on an OSCORE Group Manager, each in combination with a numeric identifier. These numeric identifiers are used to express authorization information about performing administrative operations concerning OSCORE groups under the control of the Group Manager, as specified in Section 3 of [RFC-XXXX].

The columns of this registry are:

* Name: A value that can be used in documents for easier comprehension, to identify a possible permission that Administrators can perform when interacting with an OSCORE Group Manager.

* Value: The numeric identifier for this permission. Integer values greater than 65535 are marked as "Private Use", all other values use the registration policy "Expert Review" (see Section 4.5 of [RFC8126]).

  Note that, in general, a single permission can be associated with multiple different operations that are possible to be performed when interacting with the Group Manager.

* Description: This field contains a brief description of the permission.

* Reference: This contains a pointer to the public specification for the permission.

This registry will be initially populated by the values in Table 1. The Reference column for all of these entries will be [RFC-XXXX].

10.5. Expert Review Instructions

The IANA registry established in this document is defined as "Expert Review". This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason, so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

* Clarity and correctness of registrations. Experts are expected to check the clarity of purpose and use of the requested entries. Experts should inspect the entry for the considered permission, to verify the correctness of its description against the permission as intended in the specification that defined it. Expert should consider requesting an opinion on the correctness of registered parameters from the Authentication and Authorization for Constrained Environments (ACE) Working Group and the Constrained RESTful Environments (CoRE) Working Group.

  Entries that do not meet these objective of clarity and completeness should not be registered.

* Duplicated registration and point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments.

* Experts should take into account the expected usage of permissions when approving point assignment. Given a 'Value' V as code point, the length of the encoding of (2^(V+1) - 1) should be weighed against the usage of the entry, considering the resources and capabilities of devices it will be used on. Additionally, given a 'Value' V as code point, the length of the encoding of (2^(V+1) - 1) should be weighed against how many code points resulting in that encoding length are left, and the resources and capabilities of devices it will be used on.

* Specifications are recommended. When specifications are not provided, the description provided needs to have sufficient information to verify the points above.

11. References

11.1. Normative References

[COSE.Algorithms]
IANA, "COSE Algorithms", <https://www.iana.org/assignments/cose/cose.xhtml#algorithms>.

[I-D.ietf-ace-key-groupcomm]
Palombini, F. and M. Tiloca, "Key Provisioning for Group Communication using ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-18, 16 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-18>.

[I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-oscore-16, 6 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-oscore-16>.

[I-D.ietf-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-ietf-core-groupcomm-bis-10, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-groupcomm-bis-10>.

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

RESTful Environments (Group OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-groupcomm-20, 2 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-groupcomm-20>.

[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>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/rfc/rfc6690>.

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

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

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/rfc/rfc7641>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017, <https://www.rfc-editor.org/rfc/rfc8132>.

[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>.

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

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

[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>.

[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>.

[RFC9237] Bormann, C., "An Authorization Information Format (AIF) for Authentication and Authorization for Constrained Environments (ACE)", RFC 9237, DOI 10.17487/RFC9237, August 2022, <https://www.rfc-editor.org/rfc/rfc9237>.

[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>.

[RFC9485] Bormann, C. and T. Bray, "I-Regexp: An Interoperable Regular Expression Format", RFC 9485, DOI 10.17487/RFC9485, October 2023, <https://www.rfc-editor.org/rfc/rfc9485>.

11.2. Informative References

[I-D.amsuess-core-cachable-oscore]
Amsüss, C. and M. Tiloca, "Cacheable OSCORE", Work in Progress, Internet-Draft, draft-amsuess-core-cachable-oscore-08, 10 January 2024, <https://datatracker.ietf.org/doc/html/draft-amsuess-core-cachable-oscore-08>.

[I-D.ietf-ace-revoked-token-notification]
Tiloca, M., Palombini, F., Echeverria, S., and G. Lewis, "Notification of Revoked Access Tokens in the Authentication and Authorization for Constrained Environments (ACE) Framework", Work in Progress, Internet-Draft, draft-ietf-ace-revoked-token-notification-06, 2 June 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-revoked-token-notification-06>.

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-09, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-09>.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsüss, C., and P. Van der Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-discovery-14, 8 September 2023, <https://datatracker.ietf.org/doc/html/draft-tiloca-core-oscore-discovery-14>.

[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>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/rfc/rfc7959>.

[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>.

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

[RFC9176] Amsüss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <https://www.rfc-editor.org/rfc/rfc9176>.

[RFC9177] Boucadair, M. and J. Shallow, "Constrained Application Protocol (CoAP) Block-Wise Transfer Options Supporting Robust Transmission", RFC 9177, DOI 10.17487/RFC9177, March 2022, <https://www.rfc-editor.org/rfc/rfc9177>.

Appendix A. Processing of Group Name Patterns at the AS

When processing an Authorization Request from an Administrator (see Section 4), the AS builds the authorization information expressing granted permissions as scope entries, according to the AIF data model AIF-OSCORE-GROUPCOMM and to its extension specified in Section 3. These scope entries are in turn specified as value of the 'scope' claim to include in the Access Token.

In order to evaluate the requested permissions against the access policies pertaining to the Administrator for the Group Manager in question, the AS can perform the following steps.

The following specifically refers only to "admin scope entries", i.e., scope entries that express authorization information for Administrators of OSCORE groups.

1. The AS initializes three empty sets of scope entries, namely S1, S2, and S3.

2. For each scope entry E in the 'scope' parameter of the Authorization Request, the AS performs the following actions.

   * In its access policies related to administrative operations at the Group Manager for the Administrator, the AS determines every group name superpattern P*, such that every group name matching with the pattern P of the scope entry E matches also with P*.

   * If no superpatterns are found, the AS proceeds with the next scope entry, if any. Otherwise, the AS computes Tperm* as the union of the permission sets associated with the superpatterns found at the previous step. That is, Tperm* is the inclusive OR of the binary representations of the Tperm values associated with the found superpatterns and encoding the corresponding permission sets as per Section 3.

   * The AS adds to the set S1 a scope entry, such that its Toid is the same as in the scope entry E, while its Tperm is the AND of Tperm* with the Tperm in the scope entry E.

3. For each scope entry E in the 'scope' parameter of the Authorization Request, the AS performs the following actions.

   * In its access policies related to administrative operations at the Group Manager for the Administrator, the AS determines every group name subpattern P*, such that: i) the pattern P of the scope entry E is different from P*; and ii) every group name matching with P* also matches with P.

   * If no subpatterns are found, the AS proceeds with the next scope entry, if any. Otherwise, for each found subpattern P*, the AS adds to the set S2 a scope entry, such that its Toid is the same as in the subpattern P*, while its Tperm is the AND of the Tperm from the subpattern P* with the Tperm in the scope entry E.

4. For each scope entry E in the 'scope' parameter of the Authorization Request, the AS performs the following actions.

   * For each group name pattern P* in its access policies related to administrative operations at the Group Manager for the Administrator, the AS performs the following actions.

- The AS attempts to determine a crosspattern P** such that: i) in the previous steps, P** was not identified as a superpattern or subpattern for the pattern P of the scope entry E; ii) every group name matching with P** also matches with both P and P*.

- If no crosspattern is built, the AS proceeds with the next pattern in its access policies related to administrative operations at the Group Manager for the Administrator, if any. Otherwise, the AS adds to the set S3 a scope entry, such that its Toid is the same as in the crosspattern P**, while its Tperm is the AND of the Tperm from the pattern P* and the Tperm in the scope entry E.

5. If the sets S1, S2, and S3 are all empty, the Authorization Request has not been successfully verified, and the AS returns an error response as per Section 5.8.3 of [RFC9200]. Otherwise, the AS uses the scope entries in the sets S1, S2, and S3 as the scope entries for the 'scope' claim to include in the Access Token, as per the format defined in Section 3.

Appendix B. Document Updates

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

B.1. Version -10 to -11

* Early mentioned that issued Access Tokens can have other purposes.

* Added example of array of scope entries.

* Removed moot paragraph about the benefits of group name patterns.

* Use of CBOR tag 21065 to indicate a regular expression.

* Renamed 'group_title' as 'group_description'.

* Added a second example of FETCH to the group-collection resource.

* Avoiding accidental deactivation of a group when updating it.

* Avoid alternative ways to create a group configuration resource.

* RFC 9290 is used instead of the custom format for error responses.

* Added integer abbreviations for the new entries in the "ACE Groupcomm Parameters" registry.

* Editorial fixes and improvements.

B.2. Version -09 to -10

* Consistent use of 4.03 (Forbidden) error responses.

* Removed moot, remnant statements from when CoRAL was used.

* Clarified how the Group Manager may attempt to determine an alternative group name upon creating a new group.

* Made explicit what parameters cannot change when overwriting a group configuration.

* Improved guidelines to the Group Manager on selecting an alternative group name.

* Improved guidelines to Administrators receiving an error after trying to create a new group.

* Changed description of error 11 from "No available group names" to "Unable to determine a group name".

* Editorial fixes and improvements.

B.3. Version -08 to -09

* Removed use of CoRAL.

* Use of the pairwise mode changed to true by default.

* Clarified effects on group members after a group configuration change.

* Renamed "Signature Encryption Algorithm" to "Group Encryption Algorithm".

* Renamed "sign_enc_alg" to "gp_enc_alg".

* Fixes and editorial improvements.

B.4. Version -07 to -08

* Consistency of parameter names.

* More details on consistency of message payload.

* New section on multiple, concurrent Administrators.

* Specified atomicity of write operations.

* Clarified effects of configuration overwriting on group members.

* New ACE Groupcomm Error on unsupported configuration.

* Possible reason to deviate from default parameter values.

* Added security considerations.

* CoRAL examples use CBOR diagnostic notation and Packed CBOR.

* Various clarifications and editorial improvements.

B.5. Version -06 to -07

* Alignment with renaming in draft-ietf-ace-key-groupcomm.

* Updated signaling of semantics for binary encoded scopes.

* Split between parameter registration and their CBOR abbreviations.

* Classified parameters as must/should/may be supported.

* New error code "No available group names" and related guidelines.

* Fixes in the examples.

* Editorial improvements.

B.6. Version -05 to -06

* Use and extend the same AIF data model AIF-OSCORE-GROUPCOMM defined in [I-D.ietf-ace-key-groupcomm-oscore].

* Revised Client-AS interaction, based on the used AIF data model.

* Categorized operations at the Group Manager as required and optional to support.

* Added status parameter 'gid_reuse', on reassigning OSCORE Group IDs upon group rekeying.

* Clarifications on the group name ultimately chosen by the Group Manager.

* Moved the detailed processing of group name patterns at the AS to an Appendix, as an example.

* Editorial improvements.

B.7. Version -04 to -05

* Defined format of scope based on a new AIF data model.
* Specified authorization checks at the Group Manager.
* Revised resource handlers based on the new scope format.
* Renamed 'pub_key_enc' to 'cred_fmt'.
* Mandatory to include 'group_name' in the group creation request.
* Suggesting a used 'group_name' results in a new name, not in an error.
* Distinction between authentication credentials and public keys.
* More details on informing group members about changes in the group configuration.
* Revised order of sections; editorial improvements.

B.8. Version -03 to -04

* Clarifications on what to do in case of enhanced error responses.
* Clarifications on handling default values for group parameters.
* New configuration parameters to support OSCORE deterministic requests.
* IANA considerations - Use RFC8126 terminology.
* Author's change of address.
* Editorial improvements.

B.9. Version -02 to -03

* Aligned new and old parameters to core-groupcomm-oscore and ace-key-groupcomm-oscore.
* Removed 'cs_key_params' and 'ecdh_key_params' to avoid redundant COSE capabilities of key types, consistently with draft-ietf-ace-key-groupcomm-oscore.

* Revised examples and side effects due to parameter changes.

* New error type "Group currently active".

B.10. Version -01 to -02

* Admit multiple Administrators and limited access to admin resources.

* Early design considerations for defining the format of scope.

* Additional error handling, using also error types.

* Selective update of group-configuration resources with PATCH/iPATCH.

* Editorial improvements.

B.11. Version -00 to -01

* Names of application groups as status parameter.

* Parameters related to the pairwise mode of Group OSCORE.

* Defined FETCH for group-configuration resources.

* Policies on registration of links to the Resource Directory.

* Added resource type for group-configuration resources.

* Fixes, clarifications and editorial improvements.

Acknowledgments

Klaus Hartke provided substantial contribution in defining the resource model based on group collection and group configurations.

The authors sincerely thank Christian Amsüss, Carsten Bormann, Jim Schaad, Göran Selander, and Cigdem Sengul for their comments and feedback.

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

Authors' Addresses

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

Rikard Höglund
RISE AB
Isafjordsgatan 22
SE-16440 Stockholm Kista
Sweden
Email: rikard.hoglund@ri.se

Peter van der Stok
Email: stokcons@kpnmail.nl

Francesca Palombini
Ericsson AB
Torshamnsgatan 23
SE-16440 Stockholm Kista
Sweden
Email: francesca.palombini@ericsson.com

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2024

F. Palombini
Ericsson
C. Sengul
Brunel University
M. Tiloca
RISE AB

4 March 2024

Publish-Subscribe Profile for Authentication and Authorization for Constrained Environments (ACE) draft-ietf-ace-pubsub-profile-09

Abstract

This document defines an application profile of the Authentication and Authorization for Constrained Environments (ACE) framework, to enable secure group communication in the Publish-Subscribe (pub/sub) architecture for the Constrained Application Protocol (CoAP) [draft-ietf-core-coap-pubsub], where Publishers and Subscribers communicate through a Broker. This profile relies on protocol-specific transport profiles of ACE to achieve communication security, server authentication, and proof-of-possession for a key owned by the Client and bound to an OAuth 2.0 Access Token. This document specifies the provisioning and enforcement of authorization information for Clients to act as Publishers and/or Subscribers, as well as the provisioning of keying material and security parameters that Clients use for protecting their communications end-to-end through the Broker.

Note to RFC Editor: Please replace "[draft-ietf-core-coap-pubsub]" with the RFC number of that document and delete this paragraph.

Discussion Venues

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

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 5 September 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 3
1.1. Terminology 4
2. Application Profile Overview 5
3. Getting Authorisation to Join a Pub/sub security group (A) . 9
3.1. Topic Discovery at the Broker (Optional) 11
3.2. AS Discovery at the Broker (Optional) 11
3.3. KDC Discovery at the Broker (Optional) 11
3.4. Authorisation Request/Response for the KDC and the Broker 12
3.4.1. Format of Scope 13
3.5. Authorisation response 15
3.6. Token Transfer to KDC 16
4. Client Group Communication Interface at the KDC 17
4.1. Joining a Security Group 20
4.1.1. Join Request 20
4.1.2. Join Response 23
4.1.3. Join Error Handling 27
4.2. Other Group Operations through the KDC 28
4.2.1. Querying for Group Information 28
4.2.2. Updating Authentication Credentials 29

4.2.3. Removal from a Group 29
4.2.4. Rekeying a Group 29
5. PubSub Protected Communication (C) 30
5.1. Using COSE Objects To Protect The Resource Representation 30
5.2. AEAD Nonce 33
5.3. Replay Checks 34
6. Applicability to MQTT PubSub Profile 34
7. Security Considerations 35
8. IANA Considerations 35
8.1. ACE Groupcomm Key Types Registry 36
8.2. ACE Groupcomm Profiles Registry 36
8.3. CoRE Resource Type 37
8.4. AIF Media-Type Sub-Parameters 37
8.5. CoAP Content-Formats 37
8.6. TLS Exporter Labels 38
9. References 38
9.1. Normative References 38
9.2. Informative References 41
Appendix A. Requirements on Application Profiles 43
Appendix B. Document Updates 47
B.1. Version -08 to -09 47
B.2. Version -07 to -08 47
B.3. Version -06 to -07 48
Acknowledgments 48
Authors' Addresses 48

1. Introduction

In a publish-subscribe (pub/sub) scenario, devices communicate via a Broker, which enables store-and-forward messaging between these devices. This effectively enables a form of group communication, where all the Publishers and Subscribers participating in the same pub/sub topic are considered members of the same group associated with that topic.

With a focus on the pub/sub architecture defined in [I-D.ietf-core-coap-pubsub] for the Constrained Application Protocol (CoAP) [RFC7252], this document defines an application profile of the Authentication and Authorization for Constrained Environments (ACE) framework [RFC9200], which enables pub/sub communication where a group of Publishers and Subscribers securely communicate through a Broker using CoAP.

Building on the message formats and processing defined in [I-D.ietf-ace-key-groupcomm], this document specifies the provisioning and enforcement of authorization information for Clients to act as Publishers and/or Subscribers at the Broker, as well as the provisioning of keying material and security parameters that Clients use for protecting end-to-end their communications via the Broker.

In order to protect the pub/sub operations at the Broker as well as the provisioning of keying material and security parameters, this profile relies on protocol-specific transport profiles of ACE (e.g., [RFC9202], [RFC9203], or [I-D.ietf-ace-edhoc-oscore-profile]) to achieve communication security, server authentication, and proof-of-possession for a key owned by the Client and bound to an OAuth 2.0 Access Token.

The content of published messages that are circulated by the Broker is protected end-to-end between the corresponding Publisher and the intended Subscribers. To this end, this profile relies on COSE [RFC9052][RFC9053] and on keying material provided to the Publishers and Subscribers participating in the same pub/sub topic. In particular, source authentication of published content is achieved by means of the corresponding Publisher signing such content with its own private key.

While this profile focuses on the pub/sub architecture for CoAP, this document also describes how it can be applicable to MQTT [MQTT-OASIS-Standard-v5]. Similar adaptations can also extend to further transport protocols and pub/sub architectures.

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 [RFC9200], and the Authorization Information Format (AIF) [RFC9237] to express authorization information. In particular, analogously to [RFC9200], terminology for entities in the architecture such as Client (C), Resource Server (RS), and Authorization Server (AS) is defined in OAuth 2.0 [RFC6749].

* The terms and concept related to the message formats and processing, specified in [I-D.ietf-ace-key-groupcomm], for provisioning and renewing keying material in group communication scenarios. These include the abbreviations REQx and OPTx denoting the numbered mandatory-to-address and optional-to-address requirements, respectively.

* The terms and concepts described in CDDL [RFC8610], CBOR [RFC8949], and COSE [RFC9052][RFC9053][RFC9338].

* The terms and concepts described in CoAP [RFC7252]. Unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol".

* The terms and concepts of pub/sub group communication with CoAP, as described in [I-D.ietf-core-coap-pubsub].

A party interested in participating in group communication as well as already participating as a group member is interchangeably denoted as "Client", "pub/sub client", or "node".

* Group: a set of nodes that share common keying material and security parameters to protect their communications with one another. That is, the term refers to a "security group". This is not to be confused with an "application group", which has relevance at the application level and whose members are in this case the Clients acting as Publishers and/or Subscribers for a topic.

Examples throughout this document are expressed in CBOR diagnostic notation without the tag and value abbreviations.

2. Application Profile Overview

This document describes how to use [RFC9200] and [I-D.ietf-ace-key-groupcomm] to perform authentication, authorization, and key distribution operations as overviewed in Section 2 of [I-D.ietf-ace-key-groupcomm], where the considered group is the security group composed of the pub/sub clients that exchange end-to-end protected content through the Broker.

Pub/sub clients communicate within their application groups, each of which is mapped to a topic. Depending on the application, a topic may consist of one or more sub-topics, which may have their own sub-topics and so on, thus forming a hierarchy. A security group SHOULD

be associated with a single application group. However, the same application group MAY be associated with multiple security groups. Further details and considerations on the mapping between the two types of groups are out of the scope of this document.

This profile considers the architecture shown in Figure 1. A Client can act as a Publisher, or a Subscriber, or both, e.g., by publishing to some topics and subscribing to others. However, for the simplicity of presentation, this profile describes Publisher and Subscriber Clients separately.

Both Publishers and Subscribers act as ACE Clients. The Broker acts as an ACE RS, and corresponds to the Dispatcher in [I-D.ietf-ace-key-groupcomm]. The Key Distribution Center (KDC) also acts as an ACE RS, and builds on what is defined for the KDC in [I-D.ietf-ace-key-groupcomm]. From a high-level point of view, the Clients interact with the KDC in order to join security groups and obtain the group keying material to protect end-to-end and verify the content published in the associated topics.

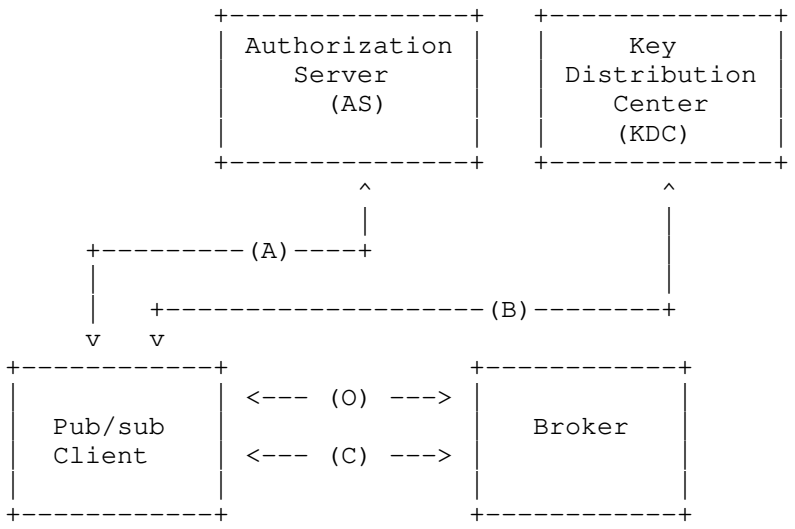


Figure 1: Architecture for Pub/sub with Authorization Server and Key Distribution Center

Both Publishers and Subscribers MUST use the same pub/sub communication protocol for their interaction with the Broker. When using the profile defined in this document, this protocol MUST be CoAP, which is used as described in [I-D.ietf-core-coap-pubsub]. What is specified in this document can apply to other pub/sub protocols such as MQTT [MQTT-OASIS-Standard-v5], or to further transport protocols.

All Publishers and Subscribers MUST use CoAP when communicating with the KDC.

Furthermore, both Publishers and Subscribers MUST use the same transport profile of ACE (e.g., [RFC9202] for DTLS; or [RFC9203] or [I-D.ietf-ace-edhoc-oscore-profile] for OSCORE) in their interaction with the Broker. In order to reduce the number of libraries that Clients have to support, it is RECOMMENDED that the same transport profile of ACE is used also for the interaction between the Clients and the KDC.

All communications between the involved entities MUST be secured.

The Client and the Broker MUST have a secure communication association, which they establish with the help of the AS and using a transport profile of ACE. This is shown by the interactions A and C in Figure 1. During this process, the Client obtains an Access Token from the AS and uploads it to the Broker, thus providing an evidence of the topics that it is authorized to participate in, and with which permissions.

The Client and the KDC MUST have a secure communication association, which they also establish with the help of the AS and using a transport profile of ACE. This is shown by the interactions A and B in Figure 1. During this process, the Client obtains an Access Token from the AS and uploads it to the KDC, thus providing an evidence of the security groups that it can join, as associated with the topics of interest at the Broker. Based on the permissions specified in the Access Token, the Client can request the KDC to join a security group, after which the Client obtains from the KDC the keying material to use for communicating with the other group members. This builds on the process for joining security groups with ACE defined in [I-D.ietf-ace-key-groupcomm] and further specified in this document.

In addition, this profile allows an anonymous Client to perform some of the discovery operations defined in Section 2.3 of [I-D.ietf-core-coap-pubsub] through the Broker, as shown by the interaction O in Figure 1. That is, an anonymous Client can discover:

* the Broker itself, by relying on the resource type "core.ps" (see Section 2.3.1 of [I-D.ietf-core-coap-pubsub]); and

* topics of interest at the Broker (i.e., the corresponding topic resources hosted at the Broker), by relying on the resource type "core.ps.conf" (see Section 2.3.2 of [I-D.ietf-core-coap-pubsub]).

However, an anonymous Client is not allowed to access topic resources at the Broker and obtain from those any additional information or metadata about the corresponding topic (e.g., the topic status, the URI of the topic-data resource where to publish or subscribe for that topic, or the URI to the KDC).

As highlighted in Figure 2, each Client maintains two different security associations pertaining to the pub/sub group communication. On the one hand, the Client has a pairwise security association with the Broker, which, as the ACE RS, verifies that the Client is authorized to perform data operations (i.e., publish, subscribe, read, delete) on a certain set of topics (Security Association 1). As discussed above, this security association is set up with the help of the AS and using a transport profile of ACE, when the Client obtains the Access Token to upload to the Broker.

On the other hand, separately for each topic, all the Publisher and Subscribers for that topic have a common, group security association, through which the published content sent through the Broker is protected end-to-end (Security Association 2). As discussed above, this security association is set up and maintained as the different Clients request the KDC to join the security group, upon which they obtain from the KDC the corresponding group keying material to use for protecting end-to-end and verifying the content of their pub/sub group communication.

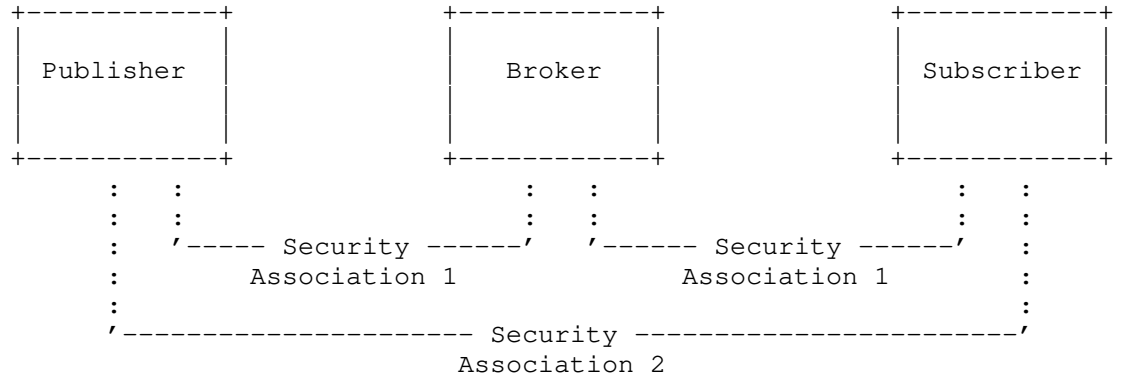


Figure 2: Security Associations between Publisher, Broker, and Subscriber.

In summary, this profile specifies the following functionalities.

1. A Client obtains the authorization to participate in a pub/sub topic at the Broker with certain permissions. This pertains operations defined in [I-D.ietf-core-coap-pubsub] for taking part in pub/sub communication with CoAP.

2. A Client obtains the authorization to join a security group with certain permissions. This allows the Client to obtain from the KDC the group keying material for communicating with other group members, i.e., to protect end-to-end and verify the content published at the Broker on topics associated with the security group.

3. A Client obtains from the KDC the authentication credentials of other group members, and provides or updates the KDC with its authentication credential.

4. A Client leaves the group or is removed from the group by the KDC.

5. The KDC renews and redistributes the group keying material (rekeying), e.g., due to a membership change in the group.

Appendix A lists the specifications on this application profile of ACE, based on the requirements defined in Appendix A of [I-D.ietf-ace-key-groupcomm].

3. Getting Authorisation to Join a Pub/sub security group (A)

Figure 3 provides a high level overview of the message flow for a Client getting authorisation to join a group. Square brackets denote optional steps. The message flow is expanded in the subsequent sections.

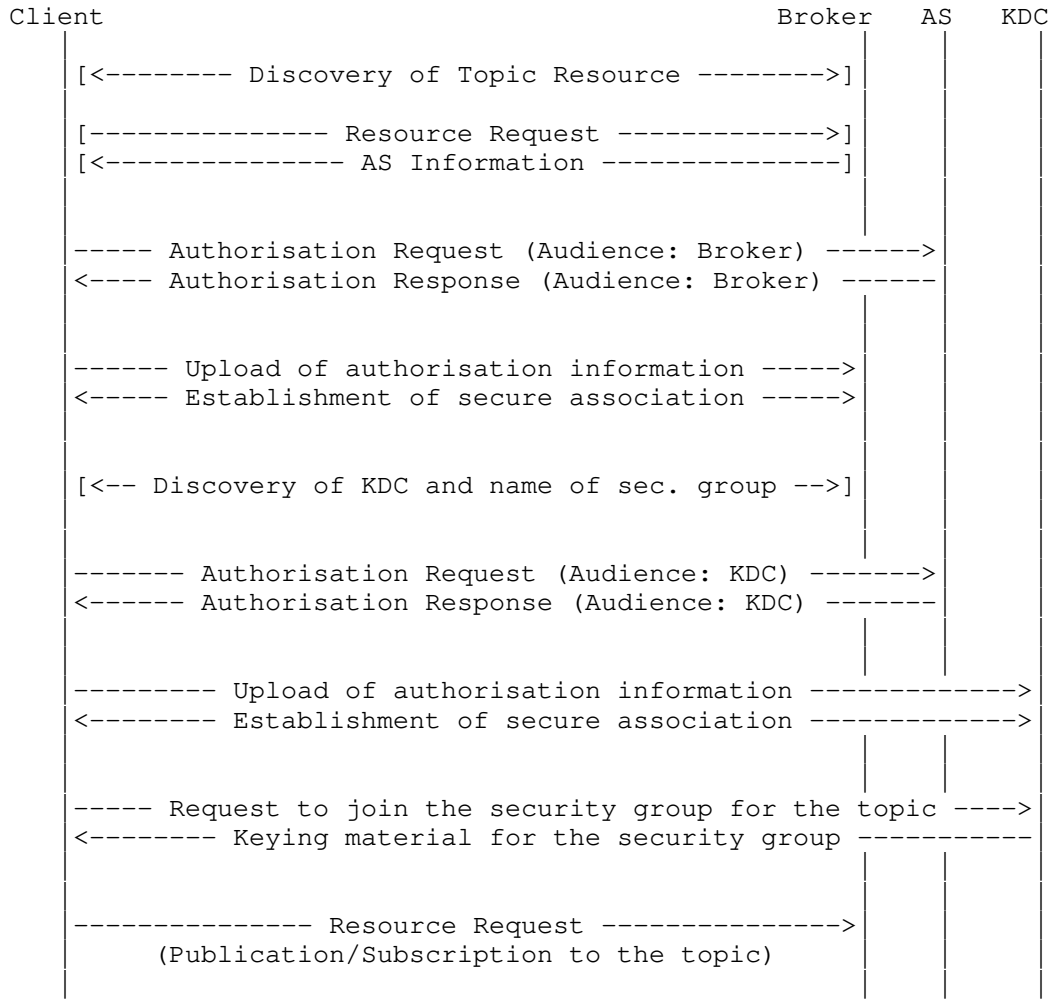


Figure 3: Authorisation Flow

Since [RFC9200] recommends the use of CoAP and CBOR, this document describes the exchanges assuming that CoAP and CBOR are used.

However, using HTTP instead of CoAP is possible, by leveraging the corresponding parameters and methods. Analogously, JSON [RFC8259] can be used instead of CBOR, using the conversion method specified in Sections 6.1 and 6.2 of [RFC8949]. In case JSON is used, the Content-Format of the message has to be changed accordingly. Exact definitions of these exchanges are out of scope for this document.

3.1. Topic Discovery at the Broker (Optional)

The discovery of a topic at the Broker can be performed by discovering the corresponding topic resources hosted at the Broker. For example, the Client can send a lookup request to /.well-known/core at the Broker, specifying as lookup criterion the resource type "core.ps.conf" (see Section 2.3.2 of [I-D.ietf-core-coap-pubsub]).

Although the links to the topic resources are also specified in the representation of the collection resource at the Broker (see Section 2.4 of [I-D.ietf-core-coap-pubsub]), the Client is not supposed to access such a resource, as intended for administrative operations that are out of the scope of this document.

3.2. AS Discovery at the Broker (Optional)

Complementary to what is defined in Section 5.1 of [RFC9200] for AS discovery, the Broker MAY send the address of the AS to the Client in the 'AS' parameter of the AS Request Creation Hints, as a response to an Unauthorized Resource Request (see Section 5.2 of [RFC9200]). An example using CBOR diagnostic notation and CoAP is given below:

```
4.01 Unauthorized
Content-Format: application/ace+cbor
Payload:
{
 "AS": "coaps://as.example.com/token"
}
```

Figure 4: AS Request Creation Hints Example

3.3. KDC Discovery at the Broker (Optional)

Once a Client has obtained an Access Token from the AS and accordingly established a secure association with the Broker, the Client has the permission to access the topic resources at the Broker that pertain to the topics on which the Client is authorized to operate.

In particular the Client is authorized to retrieve the representation of a topic resource, from which the Client can retrieve information related to the topic in question, as specified in Section 2.5 of [I-D.ietf-core-coap-pubsub].

This profile extends the set of CoAP Pub/sub Parameters that is possible to specify within the representation of a topic resource, as originally defined in Section 3 of [I-D.ietf-core-coap-pubsub]. In particular, this profile defines the following two parameters that

the Broker can specify in a response from a topic resource (see Section 2.5 of [I-D.ietf-core-coap-pubsub]). Note that, when these parameters are transported in their respective fields of the message payload, the Content-Format application/core-pubsub+cbor defined in [I-D.ietf-core-coap-pubsub] MUST be used.

* 'kdc_uri', with value the URI of the group membership resource at KDC, where Clients can send a request to join the security group associated with the topic in question. The URI is encoded as a CBOR text string. Clients will have to obtain an Access Token from the AS to upload to the KDC, before starting the process to join the security group at the KDC.

* 'sec_gp', specifying the name of the security group associated with the topic in question, as a stable and invariant identifier. The name of the security group is encoded as a CBOR text string.

Furthermore, the Resource Type (rt=) Link Target Attribute value "core.ps.gm" is registered in Section 8.3 (REQ10), and can be used to describe group-membership resources at KDC, e.g., by using a link-format document [RFC6690]. As an alternative to the discovery approach defined above and provided by the Broker, applications can use this common resource type to discover links to group-membership resources at the KDC for joining security groups associated with pub/sub topics.

3.4. Authorisation Request/Response for the KDC and the Broker

A Client sends two Authorisation Requests to the AS, targeting two different audiences, i.e., the Broker and the KDC.

As to the former, the AS handles Authorisation Requests related to a topic for which the Client is allowed to perform topic data operations at the Broker, as corresponding to an application group.

As to the latter, the AS handles Authorization Requests for security groups that the Client is allowed to join, in order to obtain the group keying material for protecting end-to-end and verifying the content of exchanged messages on the associated pub/sub topics.

This section builds on Section 3 of [I-D.ietf-ace-key-groupcomm] and defines only additions or modifications to that specification.

Both Authorisation Requests include the following fields (see Section 3.1 of [I-D.ietf-ace-key-groupcomm]):

* 'scope': Optional. If present, it specifies the following information, depending on the specifically targeted audience.

If the audience is the Broker, the scope specifies the name of the topics that the Client wishes to access, together with the corresponding permissions. If the audience is the KDC, the scope specifies the name of the security groups that the Client wishes to join, together with the corresponding permissions.

This parameter is encoded as a CBOR byte string, whose value is the binary encoding of a CBOR array. The format MUST follow the data model AIF-PUBSUB-GROUPCOMM defined in Section 3.4.1.

* 'audience': Required identifier corresponding to either the Broker or the KDC.

Other additional parameters can be included if necessary, as defined in [RFC9200].

When using this profile, it is expected that a one-to-one mapping is enforced between the application group and the security group (see Section 2). If this is not the case, the correct access policies for both sets of scopes have to be available to the AS.

3.4.1. Format of Scope

Building on Section 3.1 of [I-D.ietf-ace-key-groupcomm], this section defines the exact format and encoding of scope used in this profile.

To this end, this profile uses the Authorization Information Format (AIF) [RFC9237] (REQ1). With reference to the generic AIF model

```
AIF-Generic<Toid, Tperm> = [* [Toid, Tperm]]
```

the value of the CBOR byte string used as scope encodes the CBOR array [* [Toid, Tperm]], where each [Toid, Tperm] element corresponds to one scope entry.

Furthermore, this document defines the new AIF data model AIF-PUBSUB-GROUPCOMM that this profile MUST use to format and encode scope entries.

In particular, the following holds for each scope entry.

The object identifier ("Toid") is specialized as a CBOR item specifying the name of the groups pertaining to the scope entry.

The permission set ("Tperm") is specialized as a CBOR unsigned integer with value R, specifying the permissions that the Client wishes to have in the groups indicated by "Toid".

More specifically, the following applies when, as defined in this document, a scope entry includes as set of permissions for user-related operations performed by a pubsub Client.

* The object identifier ("Toid") is a CBOR text string, specifying the name of one application group (topic) or of the corresponding security group to which the scope entry pertains.

* The permission set ("Tperm") is a CBOR unsigned integer, whose value R specifies the operations that the Client wishes to or has been authorized to perform on the resources at the Broker associated with the application group (topic) indicated by "Toid", or on the resources at the KDC associated with the security group indicated by "Toid" (REQ1). The value R is computed as follows.

  - Each operation (i.e., permission detail) in the permission set is converted into the corresponding numeric identifier X taken from the following set.

    o Admin (0): This operation is reserved for scope entries that express permissions for Administrators of pub/sub groups.

    o AppGroup (1): This operation is signaled as wished/ authorized when "Toid" specifies the name of an application group (topic), while it is signaled as not wished/authorized when Toid specifies the name of a security group.

    o Publish (2): This operation concerns the publication of data to the topic in question, performed by means of a PUT request sent by a Publisher Client to the corresponding topic-data resource at the Broker.

    o Read (3): This operation concerns both: i) the subscription at the topic-data resource for the topic in question at the Broker, performed by means of a GET request with the CoAP Observe Option set to 0 and sent by a Subscriber Client; and ii) the simple reading of the latest data published to the topic in question, performed by means of a simple GET request sent to the same topic-data resource.

    o Delete (4): This operation concerns the deletion of the topic-data resource for the topic in question at the Broker, performed by means of a DELETE request sent to that resource.

- The set of N numeric identifiers is converted into the single value R, by taking two to the power of each numeric identifier X_1, X_2, ..., X_N, and then computing the inclusive OR of the binary representations of all the power values.

Since this application profile considers user-related operations, the "Admin" operation is signaled as not wished/authorized. That is, the scope entries MUST have the least significant bit of "Tperm" set to 0.

If the "Toid" of a scope entry in an access token specifies the name of an application group (i.e., the "AppGroup" operation is signaled as authorized), the Client has also the permission to retrieve the configuration of the application group (topic) whose name is indicated by "Toid", by sending a GET or FETCH request to the corresponding topic resource at the Broker.

The specific interactions between the Client and the Broker are defined in [I-D.ietf-core-coap-pubsub].

The following CDDL [RFC8610] notation defines a scope entry that uses the AIF-PUBSUB-GROUPCOMM data model and expresses a set of permissions.

```
AIF-PUBSUB-GROUPCOMM = AIF-Generic<pubsub-group, pubsub-perm>
 pubsub-group = tstr ; name of pub/sub topic or of
                     ; the associated security group

 pubsub-perm = uint .bits pubsub-perm-details

 pubsub-perm-details = &(
  Admin: 0,
  AppGroup: 1
  Publish: 2,
  Read: 3,
  Delete: 4
 )

 scope_entry = [pubsub-group, pubsub-perm]
```

Figure 5: Pub/sub scope using the AIF format

3.5. Authorisation response

The AS responds with an Authorization Response to each request, containing claims, as defined in Section 5.8.2 of [RFC9200] and Section 3.2 of [I-D.ietf-ace-key-groupcomm] with the following additions:

* The AS MUST include the 'expires_in' parameter. Other means for the AS to specify the lifetime of Access Tokens are out of the scope of this document.

* The AS MUST include the 'scope' parameter, when the value included in the Access Token differs from the one specified by the Client in the Authorization Request. In such a case, the second element of each scope entry specifies the set of interactions that the Client is authorized for that scope entry, encoded as specified in Section 3.4.

Furthermore, the AS MAY use the extended format of scope defined in Section 7 of [I-D.ietf-ace-key-groupcomm] for the 'scope' claim of the Access Token. In such a case, the AS MUST use the CBOR tag with tag number TAG_NUMBER, associated with the CoAP Content-Format CF_ID for the media type application/aif+cbor registered in Section 8.5 of this document (REQ28).

Note to RFC Editor: In the previous paragraph, please replace "TAG_NUMBER" with the CBOR tag number computed as TN(ct) in Section 4.3 of [RFC9277], where ct is the ID assigned to the CoAP Content-Format registered in Section 8.5 of this document. Then, please replace "CF_ID" with the ID assigned to that CoAP Content-Format. Finally, please delete this paragraph.

This indicates that the binary encoded scope follows the scope semantics defined for this application profile in Section 3.4.1 of this document.

3.6. Token Transfer to KDC

The Client transfers its access token to the KDC using one of the methods defined in the Section 3.3 of [I-D.ietf-ace-key-groupcomm]. This typically includes sending a POST request to the authz-info endpoint. However, if the DTLS transport profile of ACE [RFC9202] is used and the Client uses a symmetric proof-of-possession key in the DTLS handshake, the Client MAY provide the access token to the KDC in the "psk_identity" field of the DTLS ClientKeyExchange message when using DTLS 1.2 [RFC6347], or in the "identity" field of a PskIdentity within the PreSharedKeyExtension of the ClientHello message when using DTLS 1.3 [RFC9147]. In addition to that, the following applies.

In the Token Transfer Response to the Publishers, i.e., the Clients whose scope of the access token includes the "Publish" permission for at least one scope entry, the KDC MUST include the parameter 'kdcchallenge' in the CBOR map. 'kdcchallange' is a challenge N_S generated by the KDC, and is RECOMMENDED to be an 8-byte long random

nonce. Later when joining the group, the Publisher can use the 'kdcchallenge' as part of proving possession of its private key (see [I-D.ietf-ace-key-groupcomm]). If a Publisher provides the access token to the KDC through an authz-info endpoint, the Client MUST support the parameter 'kdcchallenge'.

If 'sign_info' is included in the Token Transfer Request, the KDC SHOULD include the 'sign_info' parameter in the Token Transfer Response. Note that the joining node may have obtained such information by alternative means e.g., the 'sign_info' may have been pre-configured (OPT3).

The following applies for each element 'sign_info_entry'.

* 'sign_alg' MUST take its value from the "Value" column of one of the recommended algorithms in the "COSE Algorithms" registry [IANA.cose_algorithms] (REQ3).

* 'sign_parameters' is a CBOR array. Its format and value are the same of the COSE capabilities array for the algorithm indicated in 'sign_alg' under the "Capabilities" column of the "COSE Algorithms" registry [IANA.cose_algorithms] (REQ4).

* 'sign_key_parameters' is a CBOR array. Its format and value are the same of the COSE capabilities array for the COSE key type of the keys used with the algorithm indicated in 'sign_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [IANA.cose_key-type] (REQ5).

* 'cred_fmt' takes value from the "Label" column of the "COSE Header Parameters" registry [IANA.cose_header-parameters] (REQ6). Acceptable values denote a format of authentication credential that MUST explicitly provide the public key as well as the comprehensive set of information related to the public key algorithm, including, e.g., the used elliptic curve (when applicable). Acceptable formats of authentication credentials include CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC7925] and C509 certificates [I-D.ietf-cose-cbor-encoded-cert]. Future formats would be acceptable to use as long as they comply with the criteria defined above.

4. Client Group Communication Interface at the KDC

In order to enable secure group communication for the pub/sub clients, the KDC provides the resources listed in Table 1. Each resource is marked as REQUIRED or OPTIONAL to be hosted at the KDC.

KDC resource	Description	Operations
/ace-group	REQUIRED. Contains a set of group names, each corresponding to one of the specified group identifiers.	FETCH (All Clients)
/ace-group/GROUPNAME	REQUIRED. Contains symmetric group keying material associated with GROUPNAME.	GET, POST (All Clients)
/ace-group/GROUPNAME/creds	REQUIRED. Contains the authentication credentials of all the Publishers of the group with name GROUPNAME.	GET, FETCH (All Clients)
/ace-group/GROUPNAME/num	REQUIRED. Contains the current version number for the symmetric group keying material of the group with name GROUPNAME.	GET (All Clients)
/ace-group/GROUPNAME/nodes/ NODENAME	REQUIRED. Contains the group keying material for	GET, DELETE (All Clients). PUT

	that group member NODENAME in GROUPNAME.	(Publishers).
/ace-group/GROUPNAME/nodes/NODENAME/cred	REQUIRED. Authentication credential for NODENAME in the group GROUPNAME.	POST (Publishers)
/ace-group/GROUPNAME/kdc-cred	REQUIRED if a group re-keying mechanism is used. Contains the authentication credential of the KDC for the group with name GROUPNAME.	GET (All Clients)
/ace-group/GROUPNAME/policies	OPTIONAL. Contains the group policies of the group with name GROUPNAME.	GET (All Clients)

Table 1: Resources at the KDC

The use of these resources follows what is defined in [I-D.ietf-ace-key-groupcomm], and only additions or modifications to that specification are defined in this document.

Consistent with what is defined in [I-D.ietf-ace-key-groupcomm], some error responses from the KDC can convey error-specific information according to the problem-details format specified in [RFC9290].

4.1. Joining a Security Group

This section describes the interactions between a Client and the KDC to join a security group. Source authentication of a message sent within the group is ensured by means of a digital signature embedded in the message. Subscribers must be able to retrieve Publishers' authentication credentials from a trusted repository, to verify source authentication of received messages. Hence, on joining a security group, a Publisher is expected to provide its own authentication credential to the KDC.

On a successful join, the Clients receive from the KDC the symmetric COSE Key used as shared group key to protect the payload of a published topic data.

The message exchange between the joining node and the KDC follows what is defined in Section 4.3.1.1 of [I-D.ietf-ace-key-groupcomm] and only additions or modifications to that specification are defined in this document.

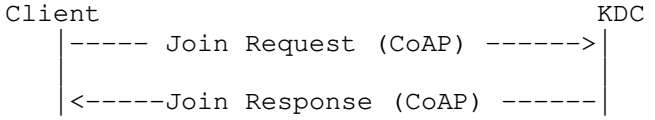


Figure 6: Join Flow

4.1.1. Join Request

After establishing a secure communication association with the KDC, the Client sends a Join Request to the KDC as described in Section 4.3 of [I-D.ietf-ace-key-groupcomm]. More specifically, the Client sends a POST request to the /ace-group/GROUPNAME endpoint, with Content-Format "application/ace-groupcomm+cbor". The payload contains the following information formatted as a CBOR map, which MUST be encoded as defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm]:

* 'scope': It MUST be present and specify the group that the Client is attempting to join, i.e., the group name, and the permissions it wishes to have in the group. This value corresponds to one scope entry, as defined in Section 3.4.1.

* 'get_creds': It MAY be present if the Client wishes to join as a Subcriber and wants to retrieve the public keys of all the Publishers upon joining. Otherwise, this parameter MUST NOT be present. If the parameter is present, the parameter MUST encode the CBOR simple value null (0xf6). Note that the parameter 'role_filter' is not necessary, as the KDC returns the authentication credentials of Publishers by default.

* 'client_cred': The use of this parameter is detailed in Section 4.1.1.1.

* 'cnonce': It specifies a dedicated nonce N_C generated by the Client. It is RECOMMENDED to use use an 8-byte long random nonce. Join Requests MUST include a new 'cnonce' at each join attempt.

* 'client_cred_verify': The use of this parameter is detailed in Section 4.1.1.2.

As a Publisher Client has its own authentication credential to use in a group, it MUST support the client_cred', 'cnonce', and 'client_cred_verify' parameters.

4.1.1.1. Client Credentials in 'client_cred'

One of the following cases can occur when a new Client attempts to join a security group.

* The joining node is not a Publisher, i.e., it is not going to send data to the application group. In this case, the joining node is not required to provide its own authentication credential to the KDC. In case the joining node still provides an authentication credential in the 'client_cred' parameter of the Join Request (see Section 4.1.1), the KDC silently ignores that parameter, as well as the related parameter 'client_cred_verify'.

* The joining node wishes to join as a Publisher, and:

  - The KDC already acquired the authentication credential of the joining node either during a past group joining process, or when establishing a secure communication association using asymmetric proof-of-possession keys. If the joining node's proof-of-possession key is compatible with the signature algorithm used in the security group and with possible associated parameters, then the corresponding authentication credential can be used in the group. In this case, the joining node MAY choose not to provide again its authentication credential to the KDC in order to limit the size of the Join Request.

- The KDC has not acquired an authentication credential. Then, the joining node MUST provide a compatible authentication credential in the 'client_cred' parameter of the Join Request (see Section 4.1.1).

Finally, the joining node MUST provide its authentication credential again if it has provided the KDC with multiple authentication credentials during past joining processes intended for different security groups. If the joining node provides its authentication credential, the KDC performs the consistency checks above and, in case of success, considers it as the authentication credential associated with the joining node in the group.

4.1.1.2. Proof-of-Possession

The 'client_cred_verify' parameter contains the proof-of-possession evidence, and is computed as defined below (REQ14).

The Publisher signs the scope, concatenated with N_S and concatenated with N_C, using the private key corresponding to the public key in the 'client_cred' parameter.

The N_S may be either:

* The challenge received from the KDC in the 'kdcchallenge' parameter of the 2.01 (Created) response to the Token Transfer Request (see Section 3.6).

* If the provisioning of the access token to the KDC has relied on the DTLS profile of ACE [RFC9202], and the access token was specified in the "psk_identity" field of the ClientKeyExchange message when using DTLS 1.2 [RFC6347], then N_S is an exporter value computed as defined in Section 4 of [RFC5705] (REQ15).

  Specifically, N_S is exported from the DTLS session between the joining node and the KDC, using an empty context value (i.e., a context value of zero-length), 32 as length value in bytes, and the exporter label "EXPORTER-ACE-Sign-Challenge-coap-group-pubsub-app" defined in Section 8.6 of this document.

* If the provisioning of the access token to the KDC has relied on the DTLS profile of ACE [RFC9202], and the access token was specified in the "identity" field of a PskIdentity within the PreSharedKeyExtension of the ClientHello message when using DTLS 1.3 [RFC9147], then N_S is an exporter value computed as defined in Section 7.5 of [RFC8446] (REQ15).

Specifically, N_S is exported from the DTLS session between the joining node and the KDC, using an empty 'context_value' (i.e., a 'context_value' of zero length), 32 as 'key_length' in bytes, and the exporter label "EXPORTER-ACE-Sign-Challenge-coap-group-pubsub-app" defined in Section 8.6 of this document.

* If the Join Request is a retry in response to an error response from the KDC, which included a new 'kdcchallenge' parameter, then N_S MUST be the new value from this parameter.

It is up to applications to define how N_S is computed in further alternative settings.

4.1.2. Join Response

On receiving the Join Request, the KDC processes the request as defined in Section 4.3.1 of [I-D.ietf-ace-key-groupcomm], and returns a success or error response.

If the 'client_cred' parameter is present, the KDC verifies the signature in the 'client_cred_verify' parameter. As PoP input, the KDC uses the value of the 'scope' parameter from the Join Request as a CBOR byte string, concatenated with N_S encoded as a CBOR byte string, concatenated with N_C encoded as a CBOR byte string. As public key of the joining node, the KDC uses either the one included in the authentication credential retrieved from the 'client_cred' parameter of the Join Request or the one from the already stored authentication credential from previous interactions with the joining node. The KDC verifies the PoP evidence, which is a signature, by using the public key of the joining node, as well as the signature algorithm used in the group and possible corresponding parameters.

In the case of any join request error, the KDC and the Client attempting to join follow the procedure defined in Section 4.1.3.

In case of success, the KDC responds with a Join Response, whose payload formatted as a CBOR map MUST contain the following fields as per Section 4.3.1 of [I-D.ietf-ace-key-groupcomm]:

* 'gkty': the key type "Group_PubSub_Keying_Material" (REQ18) registered in Section 8.1 for the 'key' parameter.

* 'key': The keying material for group communication includes the following parameters (REQ17):

  - 'cred_fmt', specifying the format of authentication credentials used in the group. This parameter takes its value from the "Label" column of the "COSE Header Parameters" registry

[IANA.cose_header-parameters]. At the time of writing this specification, acceptable formats of authentication credentials are CBOR Web Tokens (CWTs) and CWT Claims Sets (CCSs) [RFC8392], X.509 certificates [RFC7925] and C509 certificates [I-D.ietf-cose-cbor-encoded-cert]. Further formats may be available in the future, and would be acceptable to use as long as they comply with the criteria defined above (REQ6).

- 'sign_alg', specifying the Signature Algorithm used to sign messages in the group. This parameter takes values from the "Value" column of the "COSE Algorithms" registry [IANA.cose_algorithms].

- 'sign_params', specifying the parameters of the Signature Algorithm. This parameter is a CBOR array, which includes the following two elements:

  - 'sign_alg_capab' is a CBOR array, with the same format and value of the COSE capabilities array for the Signature Algorithm indicated in 'sign_alg', as specified for that algorithm in the "Capabilities" column of the "COSE Algorithms" registry [IANA.cose_algorithms].

  - 'sign_key_type_capab' is a CBOR array, with the same format and value of the COSE capabilities array for the COSE key type of the keys used with the Signature Algorithm indicated in 'sign_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [IANA.cose_key-type].

- 'group_key', specifying a COSE_Key object as defined in [RFC9052] and conveying the group key to use in the security group. The COSE_Key object MUST contain the following parameters:

  - 'kty', with value 4 (Symmetric).

  - 'alg', with value the identifier of the AEAD algorithm used in the security group.

  - 'Base IV', with value the Base Initialization Vector (Base IV) to use in the security group with this group key.

  - 'k', with value the symmetric encryption key to use as group key.

  - 'kid', with value the identifier of the COSE_Key object, hence of the group key.

This value is used as Group Identifier (Gid) of the security group, as long as the present key is used as group key in the security group.

- 'group_SenderId', specifying the Client's Sender ID encoded as a CBOR byte string. This field MUST be included if the Client is joining the security group as a Publisher, and MUST NOT be included otherwise. A Publisher Client MUST support the 'group_SenderId' parameter (REQ29).

  The Sender ID MUST be unique within the security group. The KDC MUST only assign an available Sender ID that has not been used in the security group since the last time when the current Gid value was assigned to the group (i.e., since the latest group rekeying, see Section 4.2.4). The KDC MUST NOT assign a Sender ID to the joining node if the node is not joining the group as a Publisher.

  The Sender ID can be short in length. Its maximum length in bytes is the length in bytes of the AEAD nonce for the AEAD algorithm, minus 6. This means that, when using AES-CCM-16-64-128 as AEAD algorithm in the security group, the maximum length of Sender IDs is 7 bytes.

* 'num', specifying the version number of the keying material specified in the 'key' field. The initial value of the version number MUST be set to 0 upon creating the group (REQ16).

* 'exi', which MUST be present.

* 'ace-groupcomm-profile', which MUST be present and has value "coap_group_pubsub_app" (PROFILE_TBD), which is registered in Section 8.2 (REQ19).

* 'creds', which MUST be present if the 'get_creds' parameter was present. Otherwise, it MUST NOT be present. The KDC provides the authentication credentials of all the Publishers in the security group.

* 'peer_roles', which MAY be omitted even if 'creds' is present, since each authentication credential conveyed in the 'creds' parameter: i) is associated with a Client authorized to be Publisher in the group; and ii) plays no role if such Client was also authorized to be Subscriber in the group. If 'creds' is not present, 'peer_roles' MUST NOT be present.

* 'peer_identifiers', which MUST be present if 'creds' is also present. Otherwise, it MUST NOT be present. The identifiers are the Publisher Sender IDs whose authentication credential is specified in the 'creds' parameter (REQ25).

* 'kdc_cred', which MUST be present if group re-keying is used, and is encoded as a CBOR byte string, with value the original binary representation of the KDC's authentication credential (REQ8).

* 'kdc_nonce', which MUST be present if 'kdc_cred' is present, and is encoded as a CBOR byte string, and including a dedicated nonce N_KDC generated by the KDC. For N_KDC, it is RECOMMENDED to use an 8-byte long random nonce.

* 'kdc_cred_verify', which MUST be present if 'kdc_cred' is present, and is encoded as a CBOR byte string. The KDC MUST compute the specified PoP evidence as a signature by using the signature algorithm used in the group, as well as its own private key associated with the authentication credential specified in the 'kdc_cred' parameter (REQ21).

* 'group_rekeying', which MAY be omitted if the KDC uses the "Point-to-Point" group rekeying scheme registered in Section 11.13 of [I-D.ietf-ace-key-groupcomm] as the default rekeying scheme in the group (OPT9). In any other case, the 'group_rekeying' parameter MUST be included.

After sending a successful Join Response, the KDC adds the Client to the list of current members of the security group, if that Client is not already a group member. Also, the Client is assigned a name NODENAME and a sub-resource /ace-group/GROUPNAME/nodes/NODENAME. Furthermore, the KDC associates NODENAME with the Client's access token and with the secure communication association that the KDC has with the Client. If the Client is a Publisher, its authentication credential is also associated with the tuple containing NODENAME, GROUPNAME, the current Gid, the newly assigned Publisher's Sender ID, and the Client's access token. The KDC MUST keep this association updated over time.

Note that, as long as the secure communication association between the Client and the KDC persists, the same Client re-joining the group is recognized by the KDC by virtue of such a secure communication association. As a consequence, the re-joining Client keeps the same NODENAME and the associated subresource /ace-group/GROUPNAME/nodes/NODENAME. Also, if the Client is a Publisher, it receives a new Sender ID according to the same criteria defined above.

If the application requires backward security, the KDC MUST generate updated security parameters and group keying material, and provide it to the current group members upon the new node's joining (see Section 4.2.4). In such a case, the joining node is not able to access secure communication in the pub/sub group prior its joining.

Upon receiving the Join Response, the joining node retrieves the KDC's authentication credential from the 'kdc_cred' parameter. The joining node MUST verify the proof-of-possession (PoP) evidence, which is a signature, specified in the 'kdc_cred_verify' parameter of the Join Response (REQ21).

4.1.3. Join Error Handling

The KDC MUST reply with a 4.00 (Bad Request) error response (OPT4) to the Join Request in the following cases:

* The 'client_cred' parameter is present in the Join Request and its value is not an eligible authentication credential (e.g., it is not of the format accepted in the group) (OPT8).

* The 'client_cred' parameter is present but does not include the 'cnonce' and 'client_cred_verify' parameters.

* The 'client_cred' parameter is not present while the joining node is not going to join the group exclusively as a Subscriber, and any of the following conditions holds:

  - The KDC does not store an eligible authentication credential (e.g., of the format accepted in the group) for the joining node.

  - The KDC stores multiple eligible authentication credentials (e.g., of the format accepted in the group) for the joining node.

* The 'scope' parameter is not present in the Join Request, or it is present and specifies any set of permissions not included in the list defined in Section 3.4.1.

A 4.00 (Bad Request) error response from the KDC to the joining node MAY have content format application/ace-groupcomm+cbor and contain a CBOR map as payload.

The CBOR map MAY include the 'kdcchallenge' parameter. If present, this parameter is a CBOR byte string, which encodes a newly generated 'kdcchallenge' value that the Client can use when preparing a new Join Request. In such a case, the KDC MUST store the newly generated value as the 'kdcchallenge' value associated with the joining node, which replaces the currently stored value (if any).

Upon receiving the Join Response, if 'kdc_cred' is present but the Client cannot verify the PoP evidence, the Client MUST stop processing the Join Response and MAY send a new Join Request to the KDC.

The KDC MUST return a 5.03 (Service Unavailable) response to a Client that sends a Join Request to join the security group as Publisher, in case there are currently no Sender IDs available to assign.

4.2. Other Group Operations through the KDC

4.2.1. Querying for Group Information

* '/ace-group': All Clients can send a FETCH request to retrieve a set of group names associated with their group identifiers specified in the request payload. Each element of the CBOR array 'gid' is a CBOR byte string (REQ13), which encodes the Gid of the group (see Section 4.1.2) for which the group name and the URI to the group-membership resource are provided.

* '/ace-group/GROUPNAME': All Clients can use GET requests to retrieve the symmetric group keying material of the group with the name GROUPNAME. The value of the GROUPNAME URI path and the group name in the access token scope ('gname') MUST coincide.

* '/ace-group/GROUPNAME/creds': The KDC acts as a repository of authentication credentials for the Publishers that are member of the security group with name GROUPNAME. The members of the group that are Subscribers can send GET/FETCH requests to this resource in order to retrieve the authentication credentials of all or a subset of the group members that are Publishers. The KDC silently ignores the Sender IDs included in the 'get_creds' parameter of the request that are not associated with any current Publisher group member (REQ26).

  The response from the KDC MAY omit the parameter 'peer_roles', since each authentication credential conveyed in the 'creds' parameter: i) is associated with a Client authorized to be Publisher in the group; and ii) plays no role if such Client was also authorized to be Subscriber in the group. If 'creds' is not present, 'peer_roles' MUST NOT be present.

* '/ace-group/GROUPNAME/num': All group member Clients can send a GET request to this resource in order to retrieve the current version number for the symmetric group keying material of the group with name GROUPNAME.

* '/ace-group/GROUPNAME/kdc-cred': All group member Clients can send a GET request to this resource in order to retrieve the current authentication credential of the KDC.

4.2.2. Updating Authentication Credentials

A Publisher with node name NODENAME can contact the KDC to upload a new authentication credential to use in the security group with name GROUPNAME, and replace the currently stored one. To this end, it sends a CoAP POST request to its associated sub-resource /ace-group/GROUPNAME/nodes/NODENAME/cred. The KDC replaces the stored authentication credential of this Client for the group GROUPNAME with the one specified in the request.

4.2.3. Removal from a Group

A Client with node name NODENAME can actively request to leave the security group with name GROUPNAME. In this case, the Client sends a CoAP DELETE request to the associated sub-resource /ace-group/GROUPNAME/nodes/NODENAME at the KDC. The KDC can also remove a group member due to any of the reasons described in Section 5 of [I-D.ietf-ace-key-groupcomm].

4.2.4. Rekeying a Group

The KDC MUST trigger a group rekeying as described in Section 6 of [I-D.ietf-ace-key-groupcomm] due to a change in the group membership or the current group keying material approaching its expiration time. The KDC MAY trigger regularly scheduled update of the group keying material.

Upon generating the new group keying material and before starting its distribution, the KDC MUST increment the version number of the group keying material. The KDC MUST preserve the current value of the Sender ID of each member in that group. The KDC MUST also generate a new Group Identifier (Gid) for the group, and use it as identifier of the new group key provided through the group rekeying and hereafter to Clients (re-)joining the security group (see Section 4.1.2).

The default rekeying scheme is Point-to-Point (see Section 6.1 of [I-D.ietf-ace-key-groupcomm]), where the KDC individually targets each node to rekey, using the pairwise secure communication association with that node.

If the group rekeying is performed due to one or multiple Clients joining the group as Publishers, then a rekeying message includes the Sender IDs and authentication credentials that those clients are going to use in the group. This information is specified by means of the parameters 'creds' and 'peer_identifiers', like done in the Join Response message. In the same spirit, the 'peer_roles' parameter MAY be omitted.

5. PubSub Protected Communication (C)

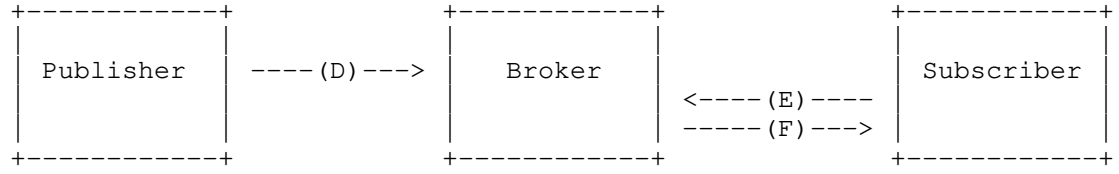


Figure 7: Secure communication between Publisher and Subscriber

(D) corresponds to the publication of a topic on the Broker, using a CoAP PUT. The publication (the resource representation) is protected with COSE ([RFC9052][RFC9053]) by the Publisher. The (E) message is the subscription of the Subscriber, and uses a CoAP GET with the Observe option set to 0 (zero) [RFC7641], as per [I-D.ietf-core-coap-pubsub]. The (F) message is the response from the Broker, where the publication is protected with COSE by the Publisher. (ToDo: Add Delete to the flow?)

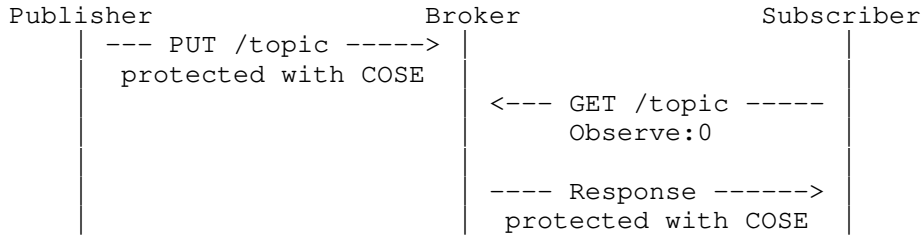


Figure 8: Example of protected communication for CoAP

5.1. Using COSE Objects To Protect The Resource Representation

The Publisher uses the symmetric COSE Key received from the KDC to protect the payload of the Publish operation (see Section 4.3 of [I-D.ietf-core-coap-pubsub]). Specifically, the Publisher creates a COSE_Encrypt0 object [RFC9052][RFC9053] by means of the COSE Key currently used as group key. The encryption algorithm and Base IV to use are specified by the 'alg' and 'Base IV' parameters of the COSE Key, together with its key identifier in the 'kid' parameter.

Also, the Publisher uses its private key corresponding to the public key sent to the KDC for countersigning the COSE_Encrypt0 object as specified in [RFC9338]. The countersignature is specified in the 'Countersignature version 2' parameter, within the 'unprotected' field of the COSE_Encrypt0 object.

Finally, the Publisher sends the COSE_Encrypt0 object conveying the countersignature to the Broker, as payload of the PUT request sent to the topic-data of the topic targeted by the Publish operation.

Upon receiving a response from the topic-data resource at the Broker, the Subscriber uses the 'kid' parameter from the 'Countersignature version 2' parameter within the 'unprotected' field of the COSE_Encrypt0 object, in order to retrieve the Publisher's public key from the Broker or from its local storage. Then, the Subscriber uses that public key to verify the countersignature.

In case of successful verification, the Subscriber uses the 'kid' parameter from the 'unprotected' field of the COSE_Encrypt0 object, in order to retrieve the COSE Key used as current group key from its local storage. Then, the Subscriber uses that group key to verify and decrypt the COSE_Encrypt0 object. In case of successful verification, the Subscriber delivers the received topic data to the application.

The COSE_Encrypt0 object is constructed as follows.

The 'protected' field MUST include:

* The 'alg' parameter, with value the identifier of the AEAD algorithm specified in the 'alg' parameter of the COSE Key used as current group key.

The 'unprotected' field MUST include:

* The 'kid' parameter, with the same value specified in the 'kid' parameter of the COSE Key used as current group key. This value represents the current Group ID (Gid) of the security group associated with the application group (topic).

* The 'Partial IV' parameter, with value set to the current Sender Sequence Number of the Publisher. All leading bytes of value zero SHALL be removed when encoding the Partial IV, except in the case of Partial IV value 0, which is encoded to the byte string 0x00.

  The Publisher MUST initialize the Sender Sequence Number to 0 upon joining the security group, and MUST reset it to 0 upon receiving a new group key as result of a group rekeying (see Section 4.2.4).

The Publisher MUST increment its Sender Sequence Number value by 1, after having completed an encryption operation by means of the current group key.

* The 'Countersignature version 2' parameter, specifying the countersignature of the COSE_Encrypt0 object. In particular:

  - The 'protected' field includes the 'alg' parameter, with value the identifier of the Signature Algorithm used in the security group.

  - The 'unprotected' field includes the 'kid' parameter, with value the Publisher's Sender ID that the Publisher obtained from the KDC when joining the security group, as value of the 'group_SenderId' parameter of the Join Response (see Section 4.1.2).

  - The 'signature' field, with value the countersignature.

  The countersignature is computed as defined in [RFC9338], by using the private key of the Publisher as signing key, and by means of the Signature Algorithm used in the group. The fields of the Countersign_structure are populated as follows:

  - 'context' takes "CounterSignature".

  - 'body_protected' takes the serialized parameters from the 'protected' field of the COSE_Encrypt0 object, i.e., the 'alg' parameter.

  - 'sign_protected' takes the serialized parameters from the 'protected' field of the 'Countersignature version 2' parameter, i.e., the 'alg' parameter.

  - 'external_aad is not supplied.

  - 'payload' is the ciphertext of the COSE_Encrypt0 object (see below).

The 'ciphertext' field specifies the ciphertext computed over the topic data to publish. The ciphertext is computed as defined in [RFC9052][RFC9053], by using the current group key as encryption key, the AEAD Nonce computed as defined in Section 5.2, the topic data to publish as plaintext, and the Enc_structure populated as follows:

* 'context' takes "Encrypt0".

* 'protected' takes the serialization of the protected parameter 'alg' from the 'protected' field of the COSE_Encrypt0 object.

* 'external_aad is not supplied.

5.2. AEAD Nonce

This section defines how to generate the AEAD nonce used for encrypting and decrypting the COSE_Encrypt0 object protecting the published topic data. This construction is analogous to that used to generate the AEAD nonce in the OSCORE security protocol (see Section 5.2 of [RFC8613]).

The AEAD nonce for producing or consuming the COSE_Encrypt0 object is constructed as defined below and also shown in Figure 9.

1. Left-pad the Partial IV (PIV) with zeroes to exactly 5 bytes.

2. Left-pad the Sender ID of the Publisher that generated the Partial IV (ID_PIV) with zeroes to exactly the nonce length of the AEAD algorithm minus 6 bytes.

3. Concatenate the size of the ID_PIV (a single byte S) with the padded ID_PIV and the padded PIV.

4. XOR the result from the previous step with the Base IV.

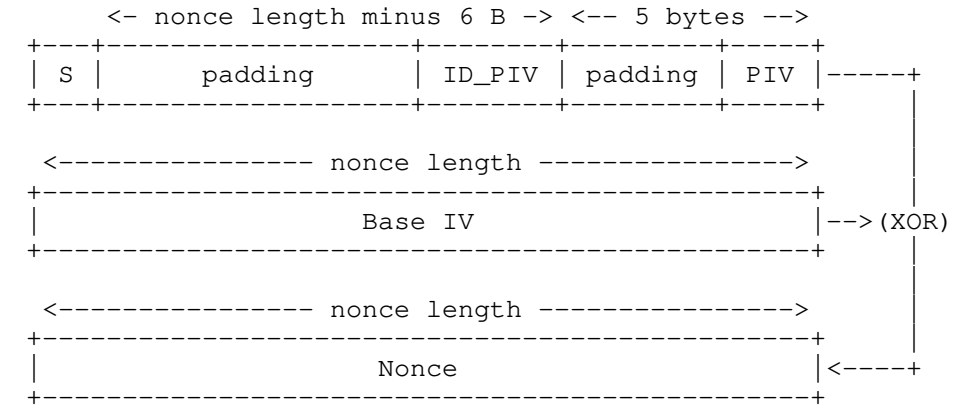


Figure 9: AEAD Nonce Construction

The construction above only supports AEAD algorithms that use nonces with length equal or greater than 7 bytes. At the same time, it makes it easy to verify that the nonces will be unique when used with the the same group key, even though this is shared and used by all the Publishers in the security group. In fact:

* Since Publisher's Sender IDs are unique within the security group and they are not reassigned until a group rekeying occurs (see Section 4.1.2 and Section 4.2.4), two Publisher Clients cannot share the same tuple (S, padded ID_PIV) by construction.

* Since a Publisher increments by 1 its Sender Sequence Number after each use that it makes of the current group key, the Publisher never reuses the same tuple (S, padded ID_PIV, padded PIV) together with the same group key.

* Therefore neither the same Publisher nor any two Publishers use the same AEAD nonce with the same group key.

5.3. Replay Checks

This section defines how a Subscriber Client checks whether the topic data conveyed in a received message from the Broker is a replay.

TBD

6. Applicability to MQTT PubSub Profile

The steps MQTT clients go through would be similar to the CoAP clients, and the payload of the MQTT PUBLISH messages will be protected using COSE. The MQTT clients needs to use CoAP to communicate to the KDC, to join security groups, and be part of the pairwise rekeying initiated by the KDC.

Authorisation Server (AS) Discovery is defined in Section 2.4.1 of [RFC9431] for MQTT v5 clients (and not supported for MQTT v3 clients). $SYS/ has been widely adopted as a prefix to topics that contain server-specific information or control APIs, and may be used for topic and KDC discovery.

When the Client sends an authorisation request to the AS using the AIF-PUBSUB-GROUPCOMM data model in the authorisation response, the 'profile' claim is set to "mqtt_pubsub_app" as defined in Section 8.2.

Both Publishers and Subscribers MUST authorise to the Broker with their respective tokens, as described in [RFC9431]. A Publisher sends PUBLISH messages for a given topic and protects the payload

with the corresponding key for the associated security group. A Subscriber may send SUBSCRIBE messages with one or multiple topic filters. A topic filter may correspond to multiple topics. The Broker forwards all PUBLISH messages to all authorised Subscribers, including the retained messages.

7. Security Considerations

All the security considerations in [I-D.ietf-ace-key-groupcomm] apply.

In the profile described above, the Publishers and Subscribers use asymmetric cryptography, which would make the message exchange quite heavy for small constrained devices. Moreover, all Subscribers must be able to access the public keys of all the Publishers to a specific topic to verify the publications.

Even though access tokens have expiration times, an access token may need to be revoked before its expiration time (see [I-D.draft-ietf-ace-revoked-token-notification] for a list of possible circumstances). Clients can be excluded from future publications through re-keying for a certain topic. This could be set up to happen on a regular basis, for certain applications. How this could be done is out of scope for this work. The method described in [I-D.draft-ietf-ace-revoked-token-notification] MAY be used to allow an Authorization Server to notify the KDC about revoked access tokens.

The Broker is only trusted with verifying that the Publisher is authorized to publish, but is not trusted with the publications itself, which it cannot read nor modify.

With respect to the reusage of nonces for Proof-of-Possession input, the same considerations apply as in the [I-D.ietf-ace-key-groupcomm-oscore].

TODO: expand on security and privacy considerations

8. IANA Considerations

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

This document has the following actions for IANA.

8.1. ACE Groupcomm Key Types Registry

IANA is asked to register the following entry in the "ACE Groupcomm Key Types" registry defined in Section 11.8 of [I-D.ietf-ace-key-groupcomm].

* Name: Group_PubSub_Keying_Material
* Key Type Value: GROUPCOMM_KEY_TBD
* Profile: coap_group_pubsub_app, defined in Section 8.2 of this document.
* Description: Encoded as described in the Section 4.1.2 of this document.
* References: [RFC9052], [RFC9053], [RFC-XXXX]

8.2. ACE Groupcomm Profiles Registry

IANA is asked to register the following entries in the "ACE Groupcomm Profiles" registry defined in Section 11.9 of [I-D.ietf-ace-key-groupcomm].

* Name: coap_group_pubsub_app
* Description: Profile for delegating client authentication and authorization for publishers and subscribers in a CoAP pub/sub setting scenario in a constrained environment.
* CBOR Value: TBD
* Reference: [RFC-XXXX]

* Name: mqtt_pubsub_app
* Description: Profile for delegating client authentication and authorization for publishers and subscribers in a MQTT pub/sub setting scenario in a constrained environment.
* CBOR Value: TBD
* Reference: [RFC-XXXX]

8.3. CoRE Resource Type

IANA is asked to register the following entry in the "Resource Type (rt=) Link Target Attribute Values" registry within the "Constrained Restful Environments (CoRE) Parameters" registry group.

* Value: "core.ps.gm"

* Description: Group-membership resource for pub/sub communication.

* Reference: [RFC-XXXX]

Clients can use this resource type to discover a group membership resource at the KDC.

8.4. AIF Media-Type Sub-Parameters

For the media-types application/aif+cbor and application/aif+json defined in Section 5.1 of [RFC9237], IANA is requested to register the following entries for the two media-type parameters Toid and Tperm, in the respective sub-registry defined in Section 5.2 of [RFC9237] within the "MIME Media Type Sub-Parameter" registry group.

* Parameter: Toid

* Name: pubsub-topic

* Description/Specification: Pub/sub topic name, corresponding to the security group

* Reference: [RFC-XXXX]

* Parameter: Tperm

* Name: pubsub-perm

* Description/Specification: Permissions corresponding to the topic data interactions in the pub/sub group

* Reference: [RFC-XXXX]

8.5. CoAP Content-Formats

IANA is asked to register the following entries to the "CoAP Content-Formats" registry within the "Constrained RESTful Environments (CoRE) Parameters" registry group.

* Content Type: application/aif+cbor;Toid="pubsub-topic",Tperm="pubsub-perm"

* Content Coding: -

* ID: 294 (suggested)

* Reference: [RFC-XXXX]

* Content Type: application/aif+json;Toid="pubsub-topic",Tperm="pubsub-perm"

* Content Coding: -

* ID: 295 (suggested)

* Reference: [RFC-XXXX]

8.6. TLS Exporter Labels

IANA is asked to register the following entry to the "TLS Exporter Labels" registry defined in Section 6 of [RFC5705] and updated in Section 12 of [RFC8447].

* Value: EXPORTER-ACE-Sign-Challenge-coap-group-pubsub-app

* DTLS-OK: Y

* Recommended: N

* Reference: [RFC-XXXX]

9. References

9.1. Normative References

[I-D.ietf-ace-key-groupcomm]
Palombini, F. and M. Tiloca, "Key Provisioning for Group Communication using ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-18, 16 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-18>.

[I-D.ietf-core-coap-pubsub]
Jimenez, J., Koster, M., and A. Keränen, "A publish-subscribe architecture for the Constrained Application Protocol (CoAP)", Work in Progress, Internet-Draft, draft-

ietf-core-coap-pubsub-13, 20 October 2023,
<https://datatracker.ietf.org/doc/html/draft-ietf-core-coap-pubsub-13>.

[IANA.cose_algorithms]
IANA, "COSE Algorithms",
<https://www.iana.org/assignments/cose>.

[IANA.cose_header-parameters]
IANA, "COSE Header Parameters",
<https://www.iana.org/assignments/cose>.

[IANA.cose_key-type]
IANA, "COSE Key Types",
<https://www.iana.org/assignments/cose>.

[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>.

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

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

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <https://www.rfc-editor.org/rfc/rfc6690>.

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

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

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/rfc/rfc7641>.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/rfc/rfc7925>.

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

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

[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>.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/rfc/rfc8447>.

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

[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>.

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

[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>.

[RFC9237] Bormann, C., "An Authorization Information Format (AIF) for Authentication and Authorization for Constrained Environments (ACE)", RFC 9237, DOI 10.17487/RFC9237, August 2022, <https://www.rfc-editor.org/rfc/rfc9237>.

[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>.

[RFC9338] Schaad, J., "CBOR Object Signing and Encryption (COSE): Countersignatures", STD 96, RFC 9338, DOI 10.17487/RFC9338, December 2022, <https://www.rfc-editor.org/rfc/rfc9338>.

9.2. Informative References

[I-D.draft-ietf-ace-revoked-token-notification]
Tiloca, M., Palombini, F., Echeverria, S., and G. Lewis, "Notification of Revoked Access Tokens in the Authentication and Authorization for Constrained Environments (ACE) Framework", Work in Progress, Internet-Draft, draft-ietf-ace-revoked-token-notification-06, 2 June 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-revoked-token-notification-06>.

[I-D.ietf-ace-edhoc-oscore-profile]
Selander, G., Mattsson, J. P., Tiloca, M., and R. Höglund, "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-03, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-edhoc-oscore-profile-03>.

[I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", Work in Progress, Internet-Draft, draft-ietf-ace-key-groupcomm-oscore-16, 6 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-ace-key-groupcomm-oscore-16>.

[I-D.ietf-cose-cbor-encoded-cert]
Mattsson, J. P., Selander, G., Raza, S., Höglund, J., and M. Furuhed, "CBOR Encoded X.509 Certificates (C509 Certificates)", Work in Progress, Internet-Draft, draft-ietf-cose-cbor-encoded-cert-09, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-09>.

[MQTT-OASIS-Standard-v5]
Banks, A., Briggs, E., Borgendale, K., and R. Gupta, "OASIS Standard MQTT Version 5.0", 2017, <http://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html>.

[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>.

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

[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>.

[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>.

Appendix A. Requirements on Application Profiles

This section lists the specifications on this profile based on the requirements defined in Appendix A of [I-D.ietf-ace-key-groupcomm].

* REQ1: Specify the format and encoding of 'scope': see Section 3.4.1.

* REQ2: If the AIF format of 'scope' is used, register its specific instance of "Toid" and "Tperm" as Media Type parameters and a corresponding Content-Format, as per the guidelines in [RFC9237]: see Section 8.4.

* REQ3: If used, specify the acceptable values for 'sign_alg': values from the "Value" column of the "COSE Algorithms" registry [IANA.cose_algorithms].

* REQ4: If used, specify the acceptable values for 'sign_parameters': format and values from the COSE algorithm capabilities as specified in the "COSE Algorithms" registry [IANA.cose_algorithms].

* REQ5: If used, specify the acceptable values for 'sign_key_parameters': its format and value are the same of the COSE capabilities array for the COSE key type of the keys used with the algorithm indicated in 'sign_alg', as specified for that key type in the "Capabilities" column of the "COSE Key Types" registry [IANA.cose_key-type].

* REQ6: Specify the acceptable formats for authentication credentials and, if used, the acceptable values for 'cred_fmt': acceptable formats explicitly provide the public key as well as the comprehensive set of information related to the public key algorithm. Takes value from the "Label" column of the "COSE Header Parameters" registry [IANA.cose_header-parameters].

* REQ7: If the value of the GROUPNAME URI path and the group name in the access token scope (gname) are not required to coincide, specify the mechanism to map the GROUPNAME value in the URI to the group name: not applicable; a perfect matching is required.

* REQ8: Define whether the KDC has an authentication credential and if this has to be provided through the 'kdc_cred' parameter: optional, see Section 4.1.2 of this document.

* REQ9: Specify if any part of the KDC interface as defined in [I-D.ietf-ace-key-groupcomm] is not supported by the KDC: some are left optional, see Section 4 of this document.

* REQ10: Register a Resource Type for the group-membership resource, which is used to discover the correct URL for sending a Join Request to the KDC: the Resource Type (rt=) Link Target Attribute value "core.ps.gm" is registered in Section Section 8.3.

* REQ11: Define what specific actions (e.g., CoAP methods) are allowed on each resource provided by the KDC interface, depending on whether the Client is a current group member; the roles that a Client is authorized to take as per the obtained access token; and the roles that the Client has as current group member: see Section 4 of this document.

* REQ12: Categorize possible newly defined operations for Clients into primary operations expected to be minimally supported and secondary operations, and provide accompanying considerations: none added.

* REQ13: Specify the encoding of group identifier: CBOR byte string, with value used also to identify the current group key used in the security group (see Section 4.1.2).

* REQ14: Specify the approaches used to compute and verify the PoP evidence to include in 'client_cred_verify', and which of those approaches is used in which case: see Section 4.1.1.2.

* REQ15: Specify how the nonce N_S is generated, if the token is not provided to the KDC through the Token Transfer Request to the authz-info endpoint (e.g., if it is used directly to validate TLS instead): see Section 4.1.1.2.

* REQ16: Define the initial value of the 'num' parameter: the initial value MUST be set to 0 (see Section 4.1.2).

* REQ17: Specify the format of the 'key' parameter and register a corresponding entry in the "ACE Groupcomm Key Types" IANA registry: see Section 4.1.2 and Section 8.1.

* REQ18: Specify the acceptable values of the 'gkty' parameter: Group_PubSub_Keying_Material, see Section 4.1.2.

* REQ19: Specify and register the application profile identifier: coap_group_pubsub_app, see Section 4.1.2 and Section 8.2.

* REQ20: If used, specify the format and content of 'group_policies' and its entries. Specify the policies default values: ToDo.

* REQ21: Specify the approaches used to compute and verify the PoP evidence to include in 'kdc_cred_verify', and which of those approaches is used in which case. If external signature verifiers are supported, specify how those provide a nonce to the KDC to be used for computing the PoP evidence: see Section 4.1.2.

* REQ22: Specify the communication protocol that the members of the group must use: CoAP [RFC7252], and for pub/sub communication [I-D.ietf-core-coap-pubsub].

* REQ23: Specify the security protocol the group members must use to protect their communication. This must provide encryption, integrity and replay protection: a symmetric COSE Key is used to create a COSE_Encrypt0 object with an AEAD algorithm specified by the KDC.

* REQ24: Specify how the communication is secured between Client and KDC. Optionally, specify transport profile of ACE [RFC9200] to use between Client and KDC: ACE transport profile such as for DTLS [RFC9202] or OSCORE [RFC9203].

* REQ25: Specify the format of the identifiers of group members: the Sender ID defined in Section 4.1.2.

* REQ26: Specify policies at the KDC to handle ids that are not included in 'get_creds': see Section 4.2.1.

* REQ27: Specify the format of newly-generated individual keying material for group members, or of the information to derive it, and corresponding CBOR label: not applicable.

* REQ28: Specify which CBOR tag is used for identifying the semantics of binary scopes, or register a new CBOR tag if a suitable one does not exist already: see Section 3.5 and Section 8.5 of this document.

* REQ29: Categorize newly defined parameters according to the same criteria of Section 8 of [I-D.ietf-ace-key-groupcomm]: a Publisher Client MUST support 'group_SenderId' in 'key'; see Section 4.1.2.

* REQ30: Define whether Clients must, should or may support the conditional parameters defined in Section 8 of [I-D.ietf-ace-key-groupcomm], and under which circumstances: a Publisher Client MUST support the client_cred', 'cnonce', and 'client_cred_verify' parameters; see Section 4.1.1. A Publisher

Client that provides the token to the KDC, through the authz-info endpoint, MUST support the parameter 'kdcchallenge'; see Section 3.6.

* OPT1: Optionally, if the textual format of 'scope' is used, specify CBOR values to use for abbreviating the role identifiers in the group: no.

* OPT2: Optionally, specify the additional parameters used in the exchange of Token Transfer Request and Response: no.

* OPT3: Optionally, specify the negotiation of parameter values for signature algorithm and signature keys, if 'sign_info' is not used: see Section 3.6.

* OPT4: Optionally, specify possible or required payload formats for specific error cases: see Section 4.1.3.

* OPT5: Optionally, specify additional identifiers of error types, as values of the 'error-id' field within the Custom Problem Detail entry 'ace-groupcomm-error': no.

* OPT6: Optionally, specify the encoding of 'creds_repo' if the default is not used: no.

* OPT7: Optionally, specify the functionalities implemented at the 'control_uri' resource hosted at the Client, including message exchange encoding and other details: no.

* OPT8: Optionally, specify the behavior of the handler in case of failure to retrieve an authentication credential for the specific node: The KDC MUST reply with a 4.00 (Bad Request) error response to the Join Request; see Section 4.1.3.

* OPT9: Optionally, define a default group rekeying scheme, to refer to in case the 'rekeying_scheme' parameter is not included in the Join Response: the "Point-to-Point" rekeying scheme registered in Section 11.12 of [I-D.ietf-ace-key-groupcomm].

* OPT10: Optionally, specify the functionalities implemented at the 'control_group_uri' resource hosted at the Client, including message exchange encoding and other details: no.

* OPT11: Optionally, specify policies that instruct Clients to retain messages and for how long, if they are unsuccessfully decrypted: no.

* OPT12: Optionally, specify for the KDC to perform group rekeying (together or instead of renewing individual keying material) when receiving a Key Renewal Request: ToDo.

* OPT13: Optionally, specify how the identifier of a group member's authentication credential is included in requests sent to other group members: no.

* OPT14: Optionally, specify additional information to include in rekeying messages for the "Point-to-Point" group rekeying scheme: ToDo.

* OPT15: Optionally, specify if Clients must or should support any of the parameters defined as optional in [I-D.ietf-ace-key-groupcomm]: no.

Appendix B. Document Updates

B.1. Version -08 to -09

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

* Improved terminology section.

* Generalized scope format for future, admin-related extensions.

* Improved definition of permissions in the format of scope.

* Clarified alternative computing of N_S Challenge when DTLS is used.

* Use of the parameter 'exi' in the Join Response.

* Use of RFC 9290 instead of the custom format of error responses.

* Fixed construction of the COSE_Encrypt0 object.

* Fixed use of the resource type "core.ps.gm".

* Updated formulation of profile requirements.

* Clarification and editorial improvements.

B.2. Version -07 to -08

* Revised presentation of the scope format.

* Revised presentation of the Join Request-Response exchange.

* The 'cnonce' parameter must be present in the Join Request.
* The 'kid' of the group key is used as Group Identifier.
* Relaxed inclusion of the 'peer_roles' parameter.
* More detailed description of the encryption and signing operations.
* Defined construction of the AEAD nonce.
* Clarifications and editorial improvements.

B.3. Version -06 to -07

* Revised abstract and introduction.
* Clarified use of "application groups".
* Revised use of protocols and transport profiles with Broker and KDC.
* Revised overview of the profile and its security associations.
* Revised presentation of authorization flow.
* Subscribers cannot be anonymous anymore.
* Revised scope definition.
* Revised Join Response.
* Revised COSE countersignature, COSE encrypt objects.
* Further clarifications, fixes and editorial improvements.

Acknowledgments

The authors wish to thank Ari Keränen, John Preuß Mattsson, Jim Schaad, Ludwig Seitz, and Göran Selander for the useful discussion and reviews that helped shape this document.

The work on this document has been partly supported by the H2020 project SIFIS-Home (Grant agreement 952652).

Authors' Addresses

Francesca Palombini
Ericsson
Email: francesca.palombini@ericsson.com

Cigdem Sengul
Brunel University
Email: csengul@acm.org

Marco Tiloca
RISE AB
Email: marco.tiloca@ri.se

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2024

R. Marin-Lopez
University of Murcia
D. Garcia-Carrillo
University of Oviedo

4 March 2024

EAP-based Authentication Service for CoAP
draft-ietf-ace-wg-coap-eap-10

Abstract

This document specifies an authentication service that uses the Extensible Authentication Protocol (EAP) transported employing Constrained Application Protocol (CoAP) messages. As such, it defines an EAP lower layer based on CoAP called CoAP-EAP. One of the main goals is to authenticate a CoAP-enabled IoT device (EAP peer) that intends to join a security domain managed by a Controller (EAP authenticator). Secondly, it allows deriving key material to protect CoAP messages exchanged between them based on Object Security for Constrained RESTful Environments (OSCORE), enabling the establishment of a security association between them.

Status of This Memo

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

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

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

This Internet-Draft will expire on 5 September 2024.

Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document.

Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 3
1.1. Requirements Language 4
2. General Architecture 4
3. CoAP-EAP Operation 5
3.1. Discovery 5
3.2. Flow of operation (OSCORE establishment) 6
3.3. Reauthentication 9
3.4. Managing the State of the Service 10
3.5. Error handling 11
3.5.1. EAP authentication failure 11
3.5.2. Non-responding endpoint 12
3.5.3. Duplicated message with /.well-known/coap-eap 12
3.6. Proxy operation in CoAP-EAP 13
4. CoAP-EAP Media type format 14
5. CBOR Objects in CoAP-EAP 14
6. Cipher suite negotiation and key derivation 15
6.1. Cipher suite negotiation 15
6.2. Deriving the OSCORE Security Context 17
7. Discussion 18
7.1. CoAP as EAP lower layer 18
7.2. Size of the EAP lower layer vs EAP method size 19
8. Security Considerations 20
8.1. Use of EAP Methods 20
8.2. Authorization 20
8.3. Allowing IP traffic connectivity to perform authentication 20
8.4. Freshness of the key material 21
8.5. Channel Binding support 21
8.6. Additional Security Considerations 21
9. IANA Considerations 22
9.1. CoAP-EAP Cipher Suites 22
9.2. CoAP-EAP Information elements 23
9.3. The Well-Known URI Registry 24
9.4. The EAP lower layer identifier registry 24
9.5. Media Types Registry 25
9.6. CoAP Content-Formats Registry 26
9.7. Resource Type (rt=) Link Target Attribute Values Registry 26
10. References 26
10.1. Normative References 26

10.2. Informative References 27
Appendix A. Flow of operation (DTLS establishment) 30
A.1. Deriving DTLS PSK and identity 32
Appendix B. Using CoAP-EAP for distributing key material for IoT networks 33
Appendix C. Examples of Use Case Scenario 33
C.1. Example 1: CoAP-EAP in ACE 34
C.2. Example 2: Multi-domain with AAA infrastructures 36
C.3. Example 3: Single domain with AAA infrastructure 36
C.4. Example 4: Single domain without AAA infrastructure . . . 36
C.5. Other use cases 36
C.5.1. CoAP-EAP for network access control 36
C.5.2. CoAP-EAP for service authentication 37
Acknowledgments 37
Authors' Addresses 37

1. Introduction

This document specifies an authentication service (application) that uses the Extensible Authentication Protocol (EAP) [RFC3748] and is built on top of the Constrained Application Protocol (CoAP)[RFC7252] called CoAP-EAP. CoAP-EAP is an application that allows authenticating two CoAP endpoints by using EAP and establishing an Object Security for Constrained RESTful Environments (OSCORE) security association between them. More specifically, this document specifies how CoAP can be used as a constrained, link-layer independent, reliable EAP lower layer [RFC3748] to transport EAP messages between a CoAP server (acting as EAP peer) and a CoAP client (acting as EAP authenticator) using CoAP messages. The CoAP client has the option of contacting a backend AAA infrastructure to complete the EAP negotiation, as described in the EAP specification [RFC3748].

EAP methods transported in CoAP MUST generate cryptographic material [RFC5247] in an MSK for this specification. The MSK is used as the basis for further cryptographic derivations. This way, CoAP messages are protected after authentication. After CoAP-EAP's operation, an OSCORE security association is established between the endpoints of the service. Using the keying material derived from the authentication, other security associations could be generated. Appendix A shows how to establish a (D)TLS security association using the keying material from the EAP authentication. One of the main applications of CoAP-EAP is Internet of Things (IoT) networks, where we can find very constrained links (e.g., limited bandwidth) and devices with limited capabilities. In these IoT scenarios, we identify the IoT device as the entity that wants to be authenticated by using EAP to join a domain that is managed by a Controller. The IoT device is in these cases the EAP peer and the Controller, the entity steering the authentication, the EAP authenticator. From now

on, the IoT device is referred to as the EAP peer and the Controller as the EAP authenticator. In these cases, EAP methods that do not require many exchanges, have short messages and use cryptographic algorithms that are manageable by constrained devices are preferable. The benefits of the EAP framework in IoT are highlighted in [EAP-framework-IoT].

1.1. Requirements Language

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 of described in CoAP [RFC7252], EAP [RFC3748] [RFC5247] and OSCORE [RFC8613].

2. General Architecture

Figure 1 illustrates the architecture defined in this document. In this architecture, the EAP peer will act as a CoAP server for this service, and the domain EAP authenticator as a CoAP client. The rationale behind this decision is that EAP requests direction is always from the EAP server to the EAP peer. Accordingly, EAP responses direction is always from the EAP peer to the EAP server.

It is worth noting that the EAP authenticator MAY interact with a backend AAA infrastructure when EAP pass-through mode is used, which will place the EAP server in the AAA server that contains the information required to authenticate the EAP peer.

The protocol stack is described in Figure 2. CoAP-EAP is an application built on top of CoAP. On top of the application, there is an EAP state machine that can run any EAP method. For this specification, the EAP method MUST be able to derive keying material. CoAP-EAP also relies on CoAP reliability mechanisms in CoAP to transport EAP: CoAP over UDP with Confirmable messages ([RFC7252]) or CoAP over TCP, TLS, or WebSockets [RFC8323].

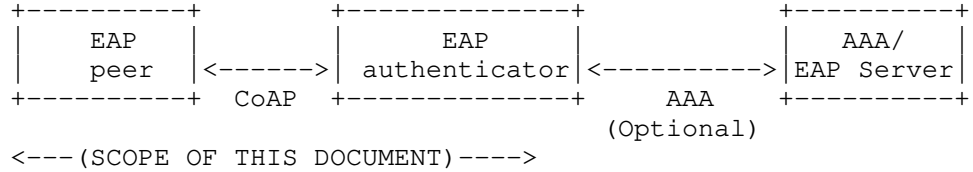


Figure 1: CoAP-EAP Architecture

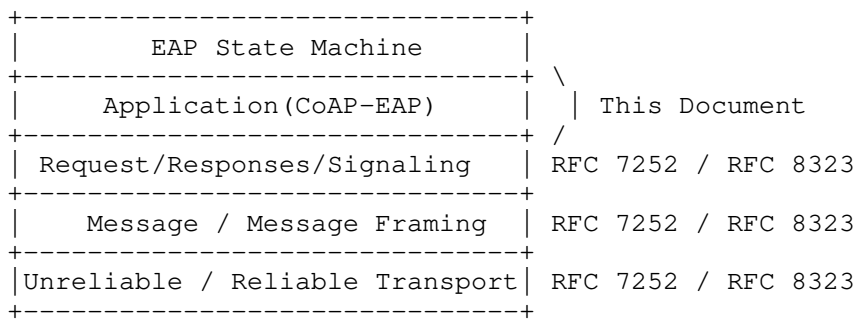


Figure 2: CoAP-EAP Stack

3. CoAP-EAP Operation

Because CoAP-EAP uses reliable delivery defined in CoAP ([RFC7252], [RFC8323]), EAP retransmission time is set to infinite as mentioned in [RFC3748]. To keep the ordering guarantee, CoAP-EAP uses Hypermedia as the Engine of Application State (HATEOAS). Each step during the EAP authentication accesses a new resource in the CoAP server (EAP peer). The previous resource is removed once the new resource is created, indicating the resource that will process the next step of the EAP authentication.

An EAP method that does not export keying material MUST NOT be used. One of the benefits of using EAP is that we can choose from a large variety of authentication methods.

In CoAP-EAP, the EAP peer will only have one authentication session with a specific EAP authenticator and it will not process any other EAP authentication in parallel (with the same EAP authenticator). That is, a single ongoing EAP authentication is permitted for the same EAP peer and the same EAP authenticator. It may be worth noting that the EAP authenticator may have parallel EAP sessions with multiple EAP peers.

To access the authentication service, this document defines the well-known URI "coap-eap" (to be assigned by IANA)

3.1. Discovery

Before the CoAP-EAP exchange takes place, the EAP peer needs to discover the EAP authenticator or the entity that will enable the CoAP-EAP exchange (e.g., an intermediary proxy). The discovery process is out of the scope of this document.

The CoAP-EAP application can be accessed through the URI "coap-eap" for the trigger message (Step 0). The CoAP-EAP service can be discovered by asking directly about the services offered. This information can also be available in the resource directory [RFC9176].

Implementation Notes: There are different methods to discover the IPv6 address of the EAP authenticator or intermediary entity. For example, on a 6LoWPAN network, the Border Router will typically act as the EAP authenticator hence, after receiving the Router Advertisement (RA) messages from the Border Router, the EAP peer may engage on the CoAP-EAP exchange. Different protocols can be used to discover the IP of the EAP authenticator. Examples of such protocols are Multicast DNS (mDNS) [RFC6762] or DHCPv6 [RFC8415].

3.2. Flow of operation (OSCORE establishment)

Figure 3 shows the general flow of operation for CoAP-EAP to authenticate using EAP and establish an OSCORE security context. The flow does not show a specific EAP method. Instead, we represent the chosen EAP method by using a generic name (EAP-X). The flow assumes that the EAP peer knows the EAP authenticator implements the CoAP-EAP service. A CoAP-EAP message has a media type application/coap-eap, See Section 9.5.

The steps of the operation are as follows:

* Step 0. The EAP peer MUST start the CoAP-EAP process by sending a "POST /.well-known/coap-eap" request (trigger message). This message carries the 'No-Response' [RFC7967] CoAP option to avoid waiting for a response that is not needed. This is the only message where the EAP authenticator acts as a CoAP server and the EAP peer as a CoAP client. The message also includes a URI in the payload of the message to indicate the resource where the EAP authenticator MUST send the next message. The name of the resource is selected by the CoAP server.

Implementation notes: When generating the URI for a resource of a step of the authentication, the resource could have the following format "path/eap/counter", where:

* "path" is some local path on the device to make the path unique. This could be omitted if desired.

* "eap" is the name that indicates the URI is for the EAP peer. This has no meaning for the protocol but helps with debugging.

* "counter' which is an incrementing unique number for every new EAP request.

So, in Figure 3 for example, the URI for the first resource would be a/eap/1"

* Step 1. The EAP authenticator sends a POST message to the resource indicated in Step 0 (e.g., '/a/eap/1'). The payload in this message contains the first EAP message (EAP Request/ Identity), the Recipient ID of the EAP authenticator (RID-C) for OSCORE, and MAY contain a CBOR array with a list of cipher suites (CS) for OSCORE. If the cipher suite list is not included, the default cipher suite for OSCORE is used. The details of the cipher suite negotiation are discussed in Section 6.1.

* Step 2. The EAP peer processes the POST message sending the EAP request (EAP-Req/Id) to the EAP peer state machine, which returns an EAP response (EAP Resp/Id). Then, assigns a new resource to the ongoing authentication process (e.g., '/a/eap/2'), and deletes the previous one ('/a/eap/1'). Finally, sends a '2.01 Created' response with the new resource identifier in the Location-Path (and/or Location-Query) options for the next step. The EAP response, the Recipient ID of the EAP peer (RID-I) and the selected cipher suite for OSCORE are included in the payload. In this step, the EAP peer MAY create an OSCORE security context (see Section 6.2). The required Master Session Key (MSK) will be available once the EAP authentication is successful in step 7.

* Steps 3-6. From now on, the EAP authenticator and the EAP peer will exchange EAP packets related to the EAP method (EAP-X), transported in the CoAP message payload. The EAP authenticator will use the POST method to send EAP requests to the EAP peer. The EAP peer will use a response to carry the EAP response in the payload. EAP requests and responses are represented in Figure 3 using the nomenclature (EAP-X-Req and EAP-X-Resp, respectively). When a POST message arrives (e.g., '/a/eap/1') carrying an EAP request message, if processed correctly by the EAP peer state machine, returns an EAP Response. Along with each EAP Response, a new resource is created (e.g., '/a/eap/3') for processing the next EAP request and the ongoing resource (e.g., '/a/eap/2') is erased. This way, ordering guarantee is achieved. Finally, an EAP response is sent in the payload of a CoAP response that will also indicate the new resource in the Location-Path (and/or Location-Query) Options. In case there is an error processing a legitimate message, the server will return a (4.00 Bad Request). There is a discussion about error handling in Section 3.5.

* Step 7. When the EAP authentication ends with success, the EAP authenticator obtains the Master Session Key (MSK) exported by the EAP method, an EAP Success message, and some authorization information (e.g., session lifetime) [RFC5247]. The EAP authenticator creates the OSCORE security context using the MSK and Recipient ID of both entities exchanged in Steps 1 and 2. The establishment of the OSCORE Security Context is defined in Section 6.2. Then, the EAP authenticator sends the POST message protected with OSCORE for key confirmation including the EAP Success. The EAP authenticator MAY also send a Session Lifetime, in seconds, which is represented with an unsigned integer in a CBOR object (see Section 5). If this Session Lifetime is not sent, the EAP peer assumes a default value of 8 hours, as RECOMMENDED in [RFC5247]. The verification of the received OSCORE-protected POST message using RID-I (Recipient ID of the EAP peer) sent in Step 2 is considered by the EAP peer as an alternate indication of success ([RFC3748]). The EAP peer state machine in the EAP peer interprets the alternate indication of success similarly to the arrival of an EAP Success and returns the MSK, which is used for the OSCORE security context in the EAP peer to process the protected POST message received from the EAP authenticator.

* Step 8. If the EAP authentication and the verification of the OSCORE-protected POST in Step 7 is successful, then the EAP peer answers with an OSCORE-protected '2.04 Changed'. From this point on, communication with the last resource (e.g., '/a/eap/(n)') MUST be protected with OSCORE. If allowed by application policy, the same OSCORE security context MAY be used to protect communication to other resources between the same endpoints.

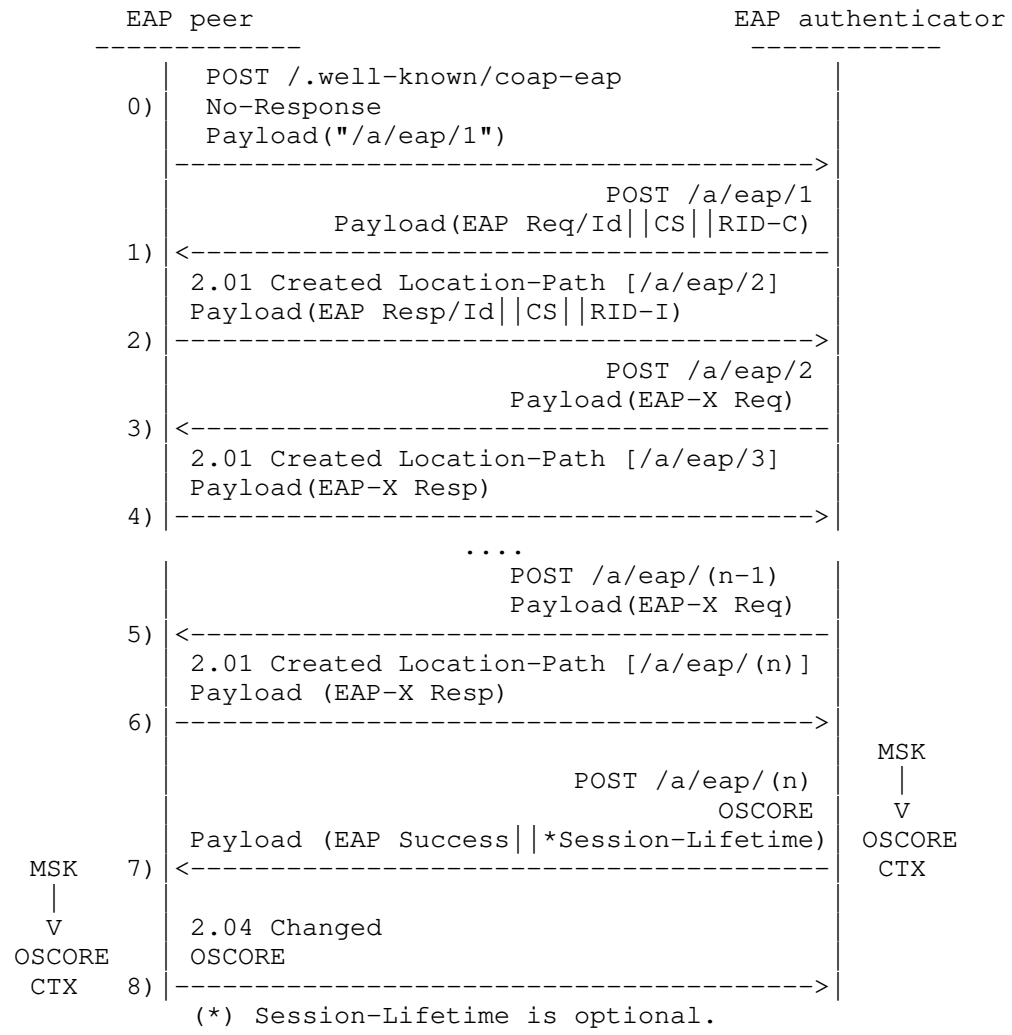


Figure 3: CoAP-EAP flow of operation with OSCORE

3.3. Reauthentication

When the CoAP-EAP state is close to expiring, the EAP peer may want to start a new authentication process (re-authentication) to renew the state. The main goal is to derive new and fresh keying material (MSK/EMSK) that, in turn, allows deriving a new OSCORE security context, increasing the protection against key leakage. The keying material MUST be renewed before the expiration of the Session-Lifetime. By default, the EAP Key Management Framework establishes a default value of 8 hours to refresh the keying material. Certain EAP

methods such as EAP-NOOB [RFC9140] or EAP-AKA' [RFC5448] provide fast reconnect for quicker re-authentication. The EAP re-authentication protocol (ERP) [RFC6696] MAY also be used to avoid the repetition of the entire EAP exchange.

The re-authentication message flow will be the same as the one shown in Figure 3. Nevertheless, two different CoAP-EAP states will be active during the re-authentication: the current CoAP-EAP state and the new CoAP-EAP state, which will be created once the re-authentication has finished with success. Once the re-authentication is completed successfully, the current CoAP-EAP state is deleted and replaced by the new CoAP-EAP state. If, for any reason, the re-authentication fails, the current CoAP-EAP state will be available until it expires, or it is renewed in another try of re-authentication.

If the re-authentication fails, it is up to the EAP peer to decide when to start a new re-authentication before the current EAP state expires.

3.4. Managing the State of the Service

The EAP peer and the EAP authenticator keep state during the CoAP-EAP negotiation. The CoAP-EAP state includes several important parts:

* A reference to an instance of the EAP (peer or authenticator/server) state machine.

* The resource for the next message in the negotiation (e.g., '/a/eap/2')

* The MSK is exported when the EAP authentication is successful. CoAP-EAP can access the different variables by the EAP state machine (i.e., [RFC4137]).

* A reference to the OSCORE context.

Once created, the EAP authenticator MAY choose to delete the state as described in Figure 4. Conversely, the EAP peer may need to renew the CoAP-EAP state because the key material is close to expiring, as mentioned in Section 3.3.

There are situations where the current CoAP-EAP state might need to be removed. For instance, due to its expiration or forced removal, the EAP peer needs to be expelled from the security domain. This exchange is illustrated in Figure 4.

If the EAP authenticator deems it necessary to remove the CoAP-EAP state from the EAP peer before it expires, it can send a DELETE command in a request to the EAP peer, referencing the last CoAP-EAP state resource given by the CoAP server, whose identifier will be the last one received (e.g., '/a/eap/(n)' in Figure 3). This message is protected by the OSCORE security association to prevent forgery. Upon reception of this message, the CoAP server sends a response to the EAP authenticator with the Code '2.02 Deleted', which is also protected by the OSCORE security association. If a response from the EAP peer does not arrive after EXCHANGE_LIFETIME the EAP authenticator will remove the state.

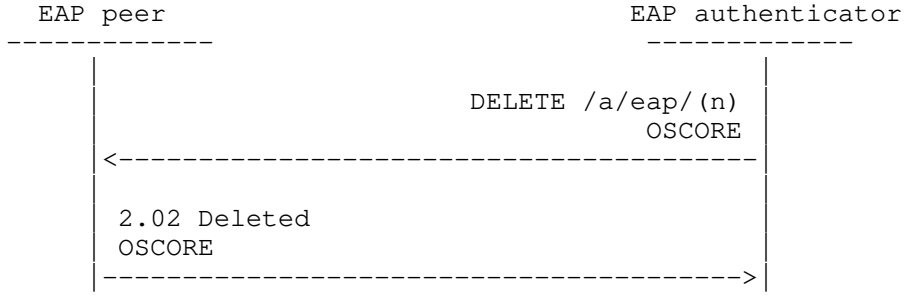


Figure 4: Deleting state

3.5. Error handling

This section elaborates on how different errors are handled. From EAP authentication failure, a non-responding endpoint lost messages, or an initial POST message arriving out of place.

3.5.1. EAP authentication failure

EAP authentication MAY fail in different situations (e.g., wrong credentials). The result is that the EAP authenticator will send an EAP Failure message because of the EAP authentication (Step 7 in Figure 3). In this case, the EAP peer MUST send a response '4.01 Unauthorized' in Step 8. Therefore, Step 7 and Step 8 are not protected in this case because no MSK is exported and the OSCORE security context is not yet generated.

If the EAP authentication fails during the re-authentication and the EAP authenticator sends an EAP failure, the current CoAP-EAP state will be still usable until it expires.

3.5.2. Non-responding endpoint

If, for any reason, one of the entities becomes non-responding, the CoAP-EAP state SHOULD be kept only for some time before it is removed. The removal of the CoAP-EAP state in the EAP authenticator assumes that the EAP peer will need to authenticate again. According to CoAP, EXCHANGE_LIFETIME considers the time it takes until a client stops expecting a response to a request. A timer is reset every time a message is sent. By default, CoAP-EAP adopts the value of EXCHANGE_LIFETIME as a timer in the EAP peer and Authenticator to remove the CoAP-EAP state if the authentication process has not progressed in that time, in consequence, it has not been completed.

The EAP peer will remove the CoAP-EAP state either if the EXCHANGE_LIFETIME is triggered, or the EAP peer state machine returns an eapFail.

The EAP authenticator will remove the CoAP-EAP state either if the EXCHANGE_LIFETIME is triggered, or, when the EAP authenticator is acting in pass-through mode (i.e., the EAP authentication is performed against a AAA server), the EAP authenticator state machine returns an aaaTimemout.

3.5.3. Duplicated message with /.well-known/coap-eap

The reception of the trigger message in Step 0 containing the URI /coap-eap needs some additional considerations, as the resource is always available in the EAP authenticator.

If a trigger message (Step 0) arrives at the EAP authenticator during an ongoing authentication with the same EAP peer, the EAP authenticator MUST silently discard this trigger message.

If an old "POST /.well-known/coap-eap" (Step 0) arrives at the EAP authenticator and there is no authentication ongoing, the EAP authenticator may understand that a new authentication process is requested. Consequently, the EAP authenticator will start a new EAP authentication. However, the EAP peer did not start any authentication and therefore, it did not select any resource for the EAP authentication. Thus, the EAP peer sends a ’4.04 Not found’ in the response (Figure 5).

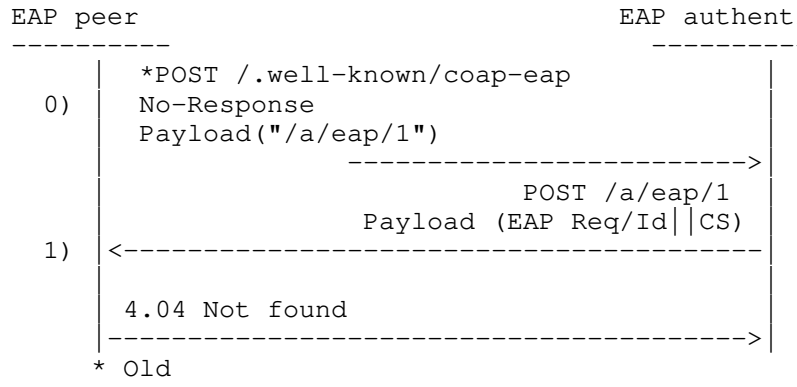


Figure 5: /.well-known/coap-eap with no ongoing authentication from the EAP authenticator

3.6. Proxy operation in CoAP-EAP

The CoAP-EAP operation is intended to be compatible with the use of intermediary entities between the EAP peer and the EAP authenticator when direct communication is not possible. In this context, CoAP proxies can be used as enablers of the CoAP-EAP exchange.

This specification is limited to using standard CoAP [RFC7252] as well as standardized CoAP options [RFC8613]. It does not specify any addition in the form of CoAP options. This is expected to ease the integration of CoAP intermediaries in the CoAP-EAP exchange.

When using proxies in the CoAP-EAP we need to consider that the exchange contains a role-reversal process at the beginning of the exchange. In the first message, the EAP peer acts as a CoAP client and the EAP authenticator as the CoAP server. After that, in the remaining exchanges the roles are reversed, being the EAP peer, the CoAP server, and the EAP authenticator, the CoAP client. When using a proxy in the exchange, for message 0, the proxy will act as forward, and as reverse for the rest. Additionally, in the first exchange, the EAP peer as a CoAP client, sends the URI for the next CoAP message in the payload of a request. This is not the typical behavior, as URIs referring to new services/resources appear in Location-Path and/or Location-Query Options in CoAP responses. Hence, the proxy will have to treat the payload of message 0, as if it were a Location-Path Option of a CoAP response.

4. CoAP-EAP Media type format

In the CoAP-EAP exchange, we will have the following format. This is the format is specified by application/coap-eap media type, see Section 9.5.

In CoAP-EAP we have two different elements that can be sent over the payload. The first one is a relative URI. This payload will be present for the first message (0) in Figure 3.

In all the other cases, we will find and EAP message followed by the CBOR Object specified in Section 5. A visual example of the second case can be found in Figure 7.

5. CBOR Objects in CoAP-EAP

In the CoAP-EAP exchange, there is information that needs to be exchanged between the two entities. Examples of this information are the cipher suites that need to be negotiated or authorization information (Session-lifetime). There may also be a need to extend the information that has to be exchanged in the future. This section specifies the CBOR [RFC8949] data structure to exchange information between the EAP peer and the EAP authenticator in the CoAP payload.

Figure 6 shows the specification of the CBOR Object to exchange information in CoAP-EAP

```
CoAP-EAP_Info = {
      ?  1 : [+ int/tstr],   ; cipher suite
      ?  2 : bstr,           ; RID-C
      ?  3 : bstr,           ; RID-I
      ?  4 : uint            ; Session-Lifetime
  }
```

Figure 6: CBOR data structure for CoAP-EAP

The parameters contain the following information:

1. cipher suite: It contains an array including the list of the proposed or selected COSE algorithms for OSCORE. If the field is carried over a request, the meaning is the proposed cipher suite, if it is carried over a response, corresponds to the agreed-upon ciphersuites.

2. RID-I: It contains the Recipient ID of the EAP peer. The EAP authenticator uses this value as a Sender ID for its OSCORE Sender Context. The EAP peer uses this value as Recipient ID for its Recipient Context.

3. RID-C: It contains the Recipient ID of the EAP authenticator. The EAP peer uses this value as a Sender ID for its OSCORE Sender Context. The EAP authenticator uses this value as Recipient ID for its Recipient Context.

4. Session-Lifetime: Contains the time the session is valid, in seconds.

Other indexes can be used to carry additional values as needed, like specific authorization parameters.

The indexes from 65000 to 65535 are reserved for experimentation.

6. Cipher suite negotiation and key derivation

6.1. Cipher suite negotiation

OSCORE runs after the EAP authentication, using the cipher suite selected in the cipher suite negotiation (Steps 1 and 2). To negotiate the cipher suite, CoAP-EAP follows a simple approach: the EAP authenticator sends a list, in decreasing order of preference, with the identifiers of the supported cipher suites (Step 1). In the response to that message (Step 2), the EAP peer sends a response with the choice.

This list is included in the payload after the EAP message through a CBOR as explained previously. An example of how the fields are arranged in the CoAP payload can be seen in Figure 7. An example of the exchange with the cipher suite negotiation is shown in Figure 8, where can be appreciated the disposition of both EAP-Request/Identity and EAP-Response/Identity, followed by the CBOR object defined in Section 5, containing the cipher suite field for the cipher suite negotiation.

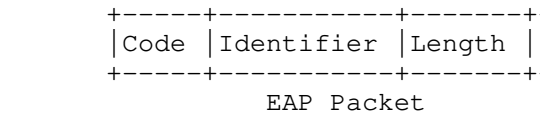


Figure 7: cipher suites are in the CoAP payload

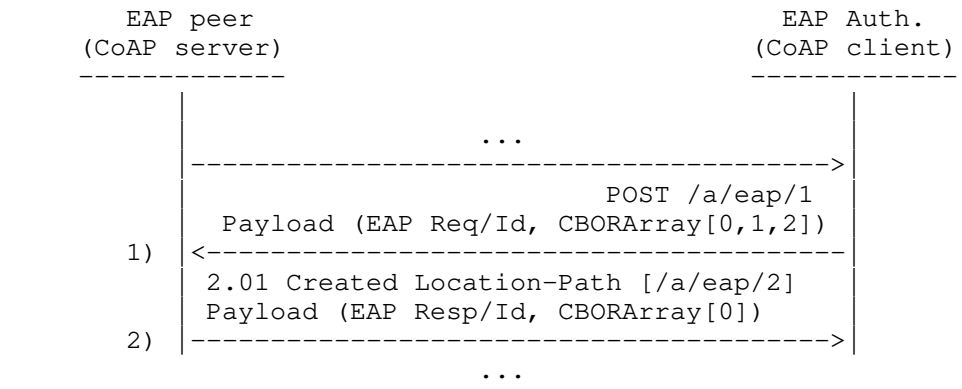


Figure 8: cipher suite negotiation

In case there is no CBOR array stating the cipher suites, the default cipher suites are applied. If the EAP authenticator sends a restricted list of cipher suites that are willing to accept, it MUST include the default value 0 since it is mandatory to implement. The EAP peer will have at least that option available.

The cipher suite requirements are inherited from the ones established by OSCORE [RFC8613], which are COSE algorithms [RFC9053]. By default, the HKDF algorithm is SHA-256 and the AEAD algorithm is AES-CCM-16-64-128 [RFC9053]. Both are mandatory to implement. The other supported and negotiated cipher suites are the following:

* 0) AES-CCM-16-64-128, SHA-256 (default)

* 1) A128GCM, SHA-256

* 2) A256GCM, SHA-384

* 3) ChaCha20/Poly1305, SHA-256

* 4) ChaCha20/Poly1305, SHAKE256

This specification uses the (HMAC)-based key derivation function (HKDF) defined in [RFC5869] to derive the necessary key material. Since the key derivation process uses the MSK, which is considered fresh key material, we will use the HKDF-Expand function, which we will shorten here as KDF. Why the use of this funtion is enough, and it is not necessary to use KDF-Extract is explained in Section 8.1.

6.2. Deriving the OSCORE Security Context

The derivation of the security context for OSCORE allows securing the communication between the EAP peer and the EAP authenticator once the MSK has been exported, providing confidentiality, integrity, key confirmation (Steps 7 and 8), and downgrading attack detection.

The Master Secret can be derived by using the chosen cipher suite and the KDF as follows:

```
Master Secret = KDF(MSK, CS | "COAP-EAP OSCORE MASTER SECRET", length)
```

where:

* The algorithms for OSCORE agreed in the cipher suite negotiation.

* The MSK exported by the EAP method. Discussion about the use of the MSK for key derivation is done in Section 8.

* CS is the concatenation of the content of the cipher suite negotiation, that is, the list of cipher suites sent by the EAP authenticator (Step 1) to the selected option by the EAP peer (Step 2). If any of the messages did not contain the CBOR array (default algorithms), the null string is used.

* "COAP-EAP OSCORE MASTER SECRET" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).

* CS and "COAP-EAP OSCORE MASTER SECRET" are concatenated.

* length is the size of the output key material.

The Master Salt, similarly to the Master Secret, can be derived as follows:

```
Master Salt = KDF(MSK, CS | "COAP-EAP OSCORE MASTER SALT", length)
```

where:

* The algorithms are agreed upon in the cipher suite negotiation.

* The MSK is exported by the EAP method. Discussion about the use of the MSK for the key derivation is done in Section 8.

* CS is the concatenation of the content of the cipher suite negotiation, in the request and response. If any of the messages did not contain the CBOR array, the null string is used.

* "COAP-EAP OSCORE MASTER SALT" is the ASCII code representation of the non-NULL-terminated string (excluding the double quotes around it).

* CS and "COAP-EAP OSCORE MASTER SALT" are concatenated.

* length is the size of the output key material.

Since the MSK is used to derive the Master Key, the correct verification of the OSCORE protected request (Step 7) and response (Step 8) confirms the EAP authenticator and the EAP peer have the same Master Secret, achieving key confirmation.

To prevent a downgrading attack, the content of the cipher suites negotiation (which we refer to here as CS) is embedded in the Master Secret derivation. If an attacker changes the value of the cipher suite negotiation, the result will be different OSCORE security contexts, which ends up with a failure in Steps 7 and 8.

The EAP authenticator will use the Recipient ID of the EAP peer (RID-I) as the Sender ID for its OSCORE Sender Context. The EAP peer will use this value as Recipient ID for its Recipient Context.

The EAP peer will use the Recipient ID of the EAP authenticator (RID-C) as the Sender ID for its OSCORE Sender Context. The EAP authenticator will use this value as Recipient ID for its Recipient Context.

7. Discussion

7.1. CoAP as EAP lower layer

This section discusses the suitability of the CoAP protocol as EAP lower layer and reviews the requisites imposed by EAP on any protocol transporting EAP. What EAP expects from its lower layers can be found in Section 3.1 of [RFC3748], which is elaborated next:

Unreliable transport. EAP does not assume that lower layers are reliable, but it can benefit from a reliable lower layer. In this sense, CoAP provides a reliability mechanism (e.g., using Confirmable messages).

Lower layer error detection. EAP relies on lower layer error detection (e.g., CRC, Checksum, MIC, etc.). CoAP goes on top of UDP/TCP, which provides a checksum mechanism over its payload.

Lower layer security. EAP does not require security services from the lower layers.

Minimum MTU. Lower layers need to provide an EAP MTU size of 1020 octets or greater. CoAP assumes an upper bound of 1024 octets for its payload, which covers the EAP requirements when in the CoAP payload only the EAP message is sent. In the case of Messages 1 and 2 in Figure 3, those messages have extra information apart from EAP. Nevertheless, the EAP Req/Id has a fixed length of 5 bytes. Message 2 with the EAP Resp/Id, would limit the length of the identity being used to the CoAP payload maximum size (1024) - len(CS || RID-I) - EAP Response header (5 bytes), which leaves enough space for sending even lengthy identities. Nevertheless, this limitation can be overcome by using CoAP block-wise transfer[RFC7959]. Note: When EAP is proxied over an AAA framework, the Access-Request packets in RADIUS MUST contain a Framed-MTU attribute with the value 1024, and the Framed-MTU AVP to 1024 in DIAMETER This attribute signals the AAA server that it should limit its responses to 1024 octets.

Ordering guarantees. EAP relies on lower layer ordering guarantees for correct operation. Regarding message ordering, every time a new message arrives at the authentication service hosted by the EAP peer, a new resource is created, and this is indicated in a "2.01 Created" response code along with the name of the new resource via Location-Path or Location-Query options. This way, the application shows that its state has advanced.

Although the [RFC3748] states: "EAP provides its own support for duplicate elimination and retransmission", EAP is also reliant on lower layer ordering guarantees. In this regard, [RFC3748] talks about possible duplication and says: "Where the lower layer is reliable, it will provide the EAP layer with a non-duplicated stream of packets. However, while it is desirable that lower layers provide for non-duplication, this is not a requirement". CoAP provides a non-duplicated stream of packets and accomplishes the desirable non-duplication. In addition, [RFC3748] says that when EAP runs over a reliable lower layer "the authenticator retransmission timer SHOULD be set to an infinite value, so that retransmissions do not occur at the EAP layer."

7.2. Size of the EAP lower layer vs EAP method size

Regarding the impact that an EAP lower layer will have on the number of bytes of the whole authentication exchange, there is a comparison with another network layer-based EAP lower layer, PANA [RFC5191], in [coap-eap].

The EAP method being transported will take most of the exchange, however, the impact of the EAP lower layer cannot be ignored, especially in very constrained communication technologies, such as the ones we can find in IoT, with limited capabilities.

Note: For constrained devices and network scenarios, the use of the latest versions of EAP methods (EAP-TLS 1.3 [RFC9190]) is recommended in favor of older versions with the goal of economization.

8. Security Considerations

There are some security aspects to be considered such as how authorization is managed, the use of MSK as key material, and how trust in the EAP authenticator is established. Additional considerations such as EAP channel binding as per [RFC6677] are also discussed here.

8.1. Use of EAP Methods

This documents limits the use of EAP methods to the ones compliant with [RFC4017] specification, yielding strong and fresh session keys such as the MSK. By this assumption, the HKDF-Expand function is used directly, as clarified in [RFC5869].

8.2. Authorization

Authorization is part of bootstrapping. It serves to establish whether the EAP peer can join and the set of conditions it must adhere to. The authorization data will be gathered from the organization that is responsible for the EAP peer and sent to the EAP authenticator in case AAA infrastructure is deployed.

In standalone mode, the authorization information will be in the EAP authenticator. If the pass-through mode is used, authorization data received from the AAA server can be delivered by the AAA protocol (e.g., RADIUS or Diameter). Providing more fine-grained authorization data can be with the transport of SAML in RADIUS [RFC7833]. After bootstrapping, additional authorization information may be needed to operate in the security domain. This can be taken care of by the solutions proposed in the ACE WG such as the use of OAuth [RFC9200], among other solutions, to provide Authentication and Authorization for Constrained Environments.

8.3. Allowing IP traffic connectivity to perform authentication

Since CoAP is an application protocol, CoAP-EAP assumes certain IP connectivity in the link between the EAP peer and the EAP authenticator to run. This link MUST authorize exclusively unprotected IP traffic to be sent between the EAP peer and the EAP authenticator during the authentication with CoAP-EAP. Once the authentication is successful, the key material generated by the EAP authentication (MSK) and any other traffic can be authorized if it is protected. It is worth noting that if the EAP authenticator is not

in the same link as the EAP peer and intermediate entity help with this process (i.e., CoAP proxy) and the same comment applies to the communication between the EAP peer and the intermediary.

8.4. Freshness of the key material

In CoAP-EAP there is no nonce exchange to provide freshness to the keys derived from the MSK. The MSK and Extended Master Session Key (EMSK) keys according to the EAP Key Management Framework [RFC5247] are fresh key material. Since only one authentication is established per EAP authenticator, there is no need to generate additional key material. In case a new MSK is required, a re-authentication can be done, by running the process again or using a more lightweight EAP method to derive additional key material as elaborated in Section 3.3.

8.5. Channel Binding support

According to the [RFC6677], channel binding related to EAP, is sent through the EAP method supporting it.

To satisfy the requirements of the document, we need to send the EAP lower layer identifier (To be assigned by IANA), in the EAP Lower-Layer Attribute if RADIUS is used.

8.6. Additional Security Considerations

In the authentication process, there is a possibility of an entity forging messages to generate denial of service (DoS) attacks on any of the entities involved. For instance, an attacker can forge multiple initial messages to start an authentication (Step 0) with the EAP authenticator as if they were sent by different EAP peers. Consequently, the EAP authenticator will start an authentication process for each message received in Step 0, sending the EAP Request/Id (Step 1).

To minimize the effects of this DoS attack, it is RECOMMENDED that the EAP authenticator limits the rate at which it processes incoming messages in Step 0 to provide robustness against denial of service (DoS) attacks. The details of rate limiting are outside the scope of this specification. Nevertheless, the rate of these messages is also limited by the bandwidth available between the EAP peer and the EAP authenticator. This bandwidth will be especially limited in constrained links (e.g., LPWAN). Lastly, it is also RECOMMENDED to reduce at a minimum the state in the EAP authenticator at least until the EAP Response/Id is received by the EAP authenticator.

Another security-related concern is how to ensure that the EAP peer joining the security domain can trust the EAP authenticator. This issue is elaborated in the EAP Key Management Framework [RFC5247]. In particular, the EAP peer knows it can trust the EAP authenticator because the key that is used to establish the security association is derived from the MSK. If the EAP authenticator has the MSK, it is because the AAA Server of the EAP peer trusted the EAP authenticator.

9. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding the registration of values related to CoAP-EAP.

9.1. CoAP-EAP Cipher Suites

IANA has created a new registry titled "CoAP-EAP Cipher Suites" under the new group name "CoAP-EAP protocol". The registration procedure is "Expert Review". The columns of the registry are Value, Array, and Description, where Value is an integer, and the other columns are text strings. The initial contents of the registry are:

* Value: -24
* Array: N/A
* Description: Reserved for Private Use
* Value: -23
* Array: N/A
* Description: Reserved for Private Use
* Value: -22
* Array: N/A
* Description: Reserved for Private Use
* Value: -21
* Array: N/A
* Description: Reserved for Private Use
* Value: 0

* Array: 10, -16
* Description: AES-CCM-16-64-128, SHA-256
* Value: 1
* Array: 1, -16
* Description: A128GCM, SHA-256
* Value: 2
* Array: 3, -43
* Description: A256GCM, SHA-384
* Value: 3
* Array: 24, -16
* Description: ChaCha20/Poly1305, SHA-256
* Value: 4
* Array: 24, -45
* Description: ChaCha20/Poly1305, SHAKE256

9.2. CoAP-EAP Information elements

IANA has created a new registry titled "CoAP-EAP Information element" under the new group name "CoAP-EAP protocol". The registration procedure is "Expert Review". The columns of the registry are Value, Name, and Description, where Value is an integer, and the other columns are text strings. The initial contents of the registry are:

* Value: 1
* Name: cipher suite
* Description: List of the proposed or selected COSE algorithms for OSCORE
* Value: 2
* Name: RID-C
* Description: It contains the Recipient ID of the EAP peer

* Value: 3

* Name: RID-I

* Description: It contains the Recipient ID of the EAP authenticator

* Value: 4

* Name: Session-Lifetime

* Description: Contains the time the session is valid in seconds

* Values: 65000 to 65535

* Name: Reserved

* Description: Reserved for experimentation

9.3. The Well-Known URI Registry

IANA has added the well-known URI "coap-eap" to the "Well-Known URIs" registry under the group name "Well-Known URIs" defined by [RFC8615]. The registration procedure is "Specification Required".

* URI suffix: coap-eap

* Change controller: IETF

* Specification document(s): [[this document]]

* Related information: None

* Status: permanent

9.4. The EAP lower layer identifier registry

IANA has added the identifier of CoAP-EAP to the registry "EAP Lower Layer" under the Extensible Authentication Protocol (EAP) Registry defined by [RFC6677]. The registration procedure is "Expert Review".

* Value: TBD.

* Lower Layer: CoAP-EAP

* Specification document(s): [[this document]]

9.5. Media Types Registry

IANA has added the media types "application/coap-eap" to the "Media Types" registry. The registration procedure is "Expert Review". Section 4 defines the format.

* Type name: application
* Subtype name: coap-eap
* Required parameters: N/A
* Optional parameters: N/A
* Encoding considerations: binary
* Security considerations: See Section 8 of this document.
* Interoperability considerations: N/A
* Published specification: [[this document]]
* Applications that use this media type: To be identified
* Fragment identifier considerations: N/A
* Additional information:
  - Magic number(s): N/A
  - File extension(s): N/A
  - Macintosh file type code(s): N/A
* Person and email address to contact for further information: See "Authors' Addresses" section.
* Intended usage: COMMON
* Restrictions on usage: N/A
* Author: See "Authors' Addresses" section.
* Change Controller: IESG

9.6. CoAP Content-Formats Registry

IANA has added the media types "application/coap-eap" to the "CoAP Content-Formats" registry under the group name "Constrained RESTful Environments (CoRE) Parameters" following the specification in Section 12.3 of [RFC7252]. The registration procedure is "IETF Review".

Media Type	Encoding	ID	Reference
application/coap-eap	-	TBD	[[this document]]

Figure 9: CoAP Content-Formats Registry

9.7. Resource Type (rt=) Link Target Attribute Values Registry

IANA has added the resource type "core.coap-eap" to the "Resource Type (rt=) Link Target Attribute Values" registry under the group name "Constrained RESTful Environments (CoRE) Parameters".

* Value: "core.coap-eap"

* Description: CoAP-EAP resource.

* Reference: [[this document]]

10. References

10.1. Normative References

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

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <https://www.rfc-editor.org/info/rfc5247>.

[RFC6677] Hartman, S., Ed., Clancy, T., and K. Hoeper, "Channel-Binding Support for Extensible Authentication Protocol (EAP) Methods", RFC 6677, DOI 10.17487/RFC6677, July 2012, <https://www.rfc-editor.org/info/rfc6677>.

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

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

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

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/info/rfc8323>.

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

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

[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/info/rfc8949>.

10.2. Informative References

[coap-eap] Garcia-Carrillo, D. and R. Marin-Lopez, "Lightweight CoAP-Based Bootstrapping Service for the Internet of Things", 2016, <https://www.mdpi.com/1424-8220/16/3/358>.

[EAP-framework-IoT]
Sethi, M., "Secure Network Access Authentication for IoT Devices: EAP Framework vs. Individual Protocols", 2021, <https://ieeexplore.ieee.org/document/9579387>.

[I-D.ingles-eap-edhoc]
Sanchez, E. I., Garcia-Carrillo, D., Marin-Lopez, R., Selander, G., and J. P. Mattsson, "Using the Extensible Authentication Protocol with Ephemeral Diffie-Hellman over COSE (EDHOC)", Work in Progress, Internet-Draft, draft-ingles-eap-edhoc-05, 12 September 2023, <https://datatracker.ietf.org/doc/html/draft-ingles-eap-edhoc-05>.

[lo-coap-eap]
Garcia-Carrillo, D., Marin-Lopez, R., Kandasamy, A., and A. Pelov, "A CoAP-Based Network Access Authentication Service for Low-Power Wide Area Networks: LO-CoAP-EAP", 2017, <https://www.mdpi.com/1424-8220/17/11/2646>.

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, DOI 10.17487/RFC4017, March 2005, <https://www.rfc-editor.org/info/rfc4017>.

[RFC4137] Vollbrecht, J., Eronen, P., Petroni, N., and Y. Ohba, "State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator", RFC 4137, DOI 10.17487/RFC4137, August 2005, <https://www.rfc-editor.org/info/rfc4137>.

[RFC4764] Bersani, F. and H. Tschofenig, "The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method", RFC 4764, DOI 10.17487/RFC4764, January 2007, <https://www.rfc-editor.org/info/rfc4764>.

[RFC5191] Forsberg, D., Ohba, Y., Ed., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, DOI 10.17487/RFC5191, May 2008, <https://www.rfc-editor.org/info/rfc5191>.

[RFC5448] Arkko, J., Lehtovirta, V., and P. Eronen, "Improved Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA')", RFC 5448, DOI 10.17487/RFC5448, May 2009, <https://www.rfc-editor.org/info/rfc5448>.

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

[RFC6696] Cao, Z., He, B., Shi, Y., Wu, Q., Ed., and G. Zorn, Ed., "EAP Extensions for the EAP Re-authentication Protocol (ERP)", RFC 6696, DOI 10.17487/RFC6696, July 2012, <https://www.rfc-editor.org/info/rfc6696>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC7833] Howlett, J., Hartman, S., and A. Perez-Mendez, Ed., "A RADIUS Attribute, Binding, Profiles, Name Identifier Format, and Confirmation Methods for the Security Assertion Markup Language (SAML)", RFC 7833, DOI 10.17487/RFC7833, May 2016, <https://www.rfc-editor.org/info/rfc7833>.

[RFC7967] Bhattacharyya, A., Bandyopadhyay, S., Pal, A., and T. Bose, "Constrained Application Protocol (CoAP) Option for No Server Response", RFC 7967, DOI 10.17487/RFC7967, August 2016, <https://www.rfc-editor.org/info/rfc7967>.

[RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/info/rfc8415>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC8824] Minaburo, A., Toutain, L., and R. Andreasen, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", RFC 8824, DOI 10.17487/RFC8824, June 2021, <https://www.rfc-editor.org/info/rfc8824>.

[RFC9031] Vuini, M., Ed., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", RFC 9031, DOI 10.17487/RFC9031, May 2021, <https://www.rfc-editor.org/info/rfc9031>.

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

[RFC9140] Aura, T., Sethi, M., and A. Peltonen, "Nimble Out-of-Band Authentication for EAP (EAP-NOOB)", RFC 9140, DOI 10.17487/RFC9140, December 2021, <https://www.rfc-editor.org/info/rfc9140>.

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

[RFC9176] Amsüss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <https://www.rfc-editor.org/info/rfc9176>.

[RFC9190] Preuß Mattsson, J. and M. Sethi, "EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3", RFC 9190, DOI 10.17487/RFC9190, February 2022, <https://www.rfc-editor.org/info/rfc9190>.

[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/info/rfc9200>.

[THREAD] Thread Group, "Thread specification 1.3", 2023.

[TS133.501] ETSI, "5G; Security architecture and procedures for 5G System - TS 133 501 V15.2.0 (2018-10)", 2018.

[ZigbeeIP] Zigbee Alliance, "ZigBee IP Specification (Zigbee Document 095023r34)", 2014.

Appendix A. Flow of operation (DTLS establishment)

CoAP-EAP makes it possible to derive a PSK from the MSK to allow (D)TLS PSK-based authentication between the EAP peer and the EAP authenticator instead of using OSCORE. In the case of using (D)TLS to establish a security association, there is a limitation on the use of intermediaries between the EAP peer and the EAP authenticator, as (D)TLS breaks the end-to-end communications when using intermediaries such as proxies.

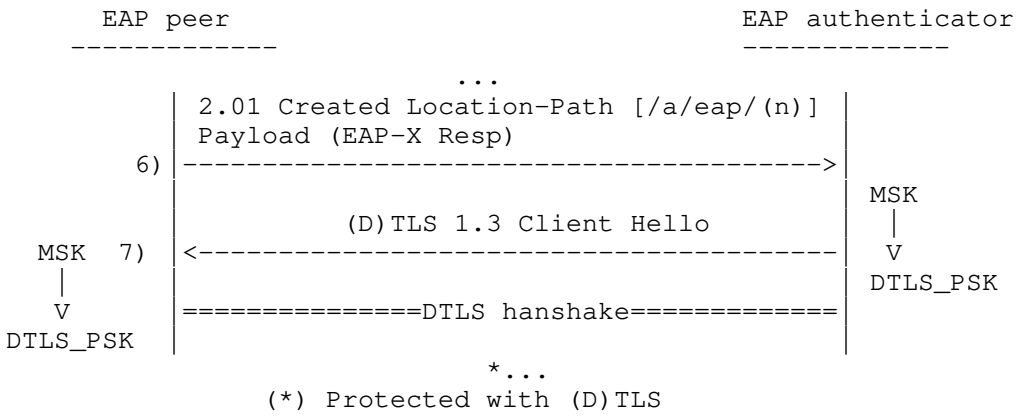


Figure 10: CoAP-EAP flow of operation with DTLS

Figure 10 shows the last steps of the operation for CoAP-EAP when (D)TLS is used to protect the communication between the EAP peer and the EAP authenticator using the keying material exported by the EAP authentication. The general flow is essentially the same as in the case of OSCORE, except that DTLS negotiation is established in Step 7). Once DTLS negotiation has finished successfully the EAP peer is granted access to the domain. Step 7 MUST be interpreted by the EAP peer as an alternate success indication, which will end up with the MSK and the DTLS_PSK derivation for the (D)TLS authentication based on PSK.

According to [RFC8446] the provision of the PSK out-of-band also requires the provision of the KDF hash algorithm and the PSK identity. To simplify the design in CoAP-EAP, the KDF hash algorithm can be included in the list of cipher suites exchanged in Step 1 and Step 2 if DTLS wants to be used instead of OSCORE. For the same reason, the PSK identity is derived from (RID-C) (RID-I) as defined in Appendix A.1.

Analogously to how the cipher suite is negotiated for OSCORE Section 6.1, the EAP authenticator sends a list, in decreasing order of preference, with the identifiers of the hash algorithms supported (Step 1). In the response, the EAP peer sends the choice.

This list is included in the payload after the EAP message with a CBOR array that contains the cipher suites. This CBOR array is enclosed as one of the elements of the CBOR Object used for transporting information in CoAP-EAP (See Section 5). An example of how the fields are arranged in the CoAP payload can be seen in Figure 7.

In case there is no CBOR array stating the cipher suites, the default cipher suites are applied. If the EAP authenticator sends a restricted list of cipher suites that is willing to accept, it MUST include the default value 0 since it is mandatory to implement. The EAP peer will have at least that option available.

For DTLS, the negotiated cipher suite is used to determine the hash function to be used to derive the DTLS PSK from the MSK:

The hash algorithms considered are the following:

* 5) TLS_SHA256

* 6) TLS_SHA384

* 7) TLS_SHA512

The inclusion of any of these values, apart from indicating the hash algorithm, indicates if the EAP authenticator intends to establish an OSCORE security association or a DTLS security association after the authentication is completed. If both options appear in the cipher suite negotiation, the OSCORE security association will be preferred over DTLS.

A.1. Deriving DTLS PSK and identity

To enable DTLS after an EAP authentication, we define the Identity and the PSK for DTLS. The Identity in this case is generated by concatenating the exchanged Sender ID and the Recipient ID.

CoAP-EAP PSK Identity = RID-C || RID-I

It is also possible to derive a pre-shared key for DTLS [RFC9147], referred to here as "DTLS PSK", from the MSK between both the EAP peer and EAP authenticator if required. The length of the DTLS PSK will depend on the cipher suite. To have keying material with sufficient length a key of 32 bytes is derived that can be later truncated if needed:

DTLS PSK = KDF(MSK, "CoAP-EAP DTLS PSK", length).

where:

* MSK is exported by the EAP method.

* "CoAP-EAP DTLS PSK" is the ASCII code representation of the non-NULL terminated string (excluding the double quotes around it).

* length is the size of the output key material.

Appendix B. Using CoAP-EAP for distributing key material for IoT networks

Similarly, to the example of Appendix A.1, where a shared key PSK for DTLS is derived, it is possible to provide key material to different link-layers after the CoAP-EAP authentication is complete.

One example is that we could use CoAP-EAP to derive the required PSK required to run the 6TiSCH Constrained Join Protocol (CoJP) [RFC9031].

Another example is when a shared Network Key is required by the devices that join a network. An example of this Network Key can be found in ZigBee IP [ZigbeeIP] or THREAD protocol [THREAD]. After CoAP-EAP execution a security association based on OSCORE protects any exchange between the EAP peer and the EAP authenticator. This security association can be used for distributing the Network Key securely and other required parameters. How the Network Key is distributed after a successful CoAP-EAP authentication is out of the scope of this document.

How a particular link-layer technology uses the MSK to derive further key material for protecting the link-layer or use the OSCORE protection to distribute key material is out of the scope of this document.

Appendix C. Examples of Use Case Scenario

In IoT, for an EAP peer to act as a trustworthy entity within a security domain, certain key material needs to be shared between the EAP peer and the EAP authenticator.

Next, we elaborate on examples of different use case scenarios about the usage of CoAP-EAP.

Generally, we are dealing with 4 entities:

* 2 EAP peers (A and B), which are EAP peers. They are the EAP peers.

* 1 EAP authenticator (C). The EAP authenticator manages a domain where EAP peers can be deployed. In IoT, it can be considered a more powerful machine than the EAP peers.

* 1 AAA server (AAA) - Optional. The AAA is an Authentication, Authorization, and Accounting Server, which is not constrained. Here, the EAP authenticator acts as an EAP authenticator in pass-through mode.

Generally, any EAP peer wanting to join the domain managed by the EAP authenticator MUST perform a CoAP-EAP authentication with the EAP authenticator (C). This authentication MAY involve an external AAA server. This means that A and B, once deployed, will run CoAP-EAP once, as a bootstrapping phase, to establish a security association with C. Moreover, any other entity, which wants to join and establish communications with EAP peers under C's domain must also do the same.

By using EAP, we can have the flexibility of having different types of credentials. For instance, if we have a device that is not battery-dependent, and not very constrained, we could use a heavier authentication method. With varied EAP peers and networks, we might need to resort to more lightweight authentication methods (e.g., EAP-NOOB[RFC9140], EAP-AKA'[RFC5448], EAP-PSK[RFC4764], EAP-EDHOC[I-D.ingles-eap-edhoc], etc.) being able to adapt to different types of devices according to organization policies or devices capabilities.

C.1. Example 1: CoAP-EAP in ACE

In ACE, the process of client registration and provisioning of credentials to the client is not specified. The process of Client registration and provisioning can be achieved using CoAP-EAP. Once the process of authentication with EAP is completed, the fresh key material is shared between the EAP peer and the EAP authenticator. In this instance, the EAP authenticator and the Authorization Server (AS) of ACE can be co-located.

Next, we exemplify how CoAP-EAP can be used to perform Client registration in a general way, to allow two EAP peers (A and B) to communicate and interact after a successful client registration.

EAP peer A wants to communicate with EAP peer B (e.g., to activate a light switch). The overall process is divided into three phases. Let's start with EAP peer A. In the first phase, EAP peer A does not yet belong to EAP authenticator C's domain. Then, it communicates with C and authenticates with CoAP-EAP, which, optionally, communicates with the AAA server to complete the authentication process. If the authentication is successful, a fresh MSK is shared between C and EAP peer A. This key material allows EAP peer A to establish a security association with the C. Some authorization information may be also provided in this step. In case EAP is used in standalone mode, the AS itself having information about the devices can be the entity providing said authorization information.

If authentication and authorization are correct, EAP peer A has been enrolled in the EAP authenticator C's domain for some time. In particular, [RFC5247] recommends 8 hours, though the entity providing the authorization information can establish this lifetime. In the same manner, B needs to perform the same process with CoAP-EAP to be part of EAP authenticator C's domain.

In the second phase, when EAP peer A wants to talk with EAP peer B, it contacts EAP authenticator C for authorization to access EAP peer B and obtain all the required information to do that securely (e.g., keys, tokens, authorization information, etc.). This phase does NOT require the usage of CoAP-EAP. The details of this phase are out of the scope of this document, and the ACE framework is used for this purpose [RFC9200].

In the third phase, EAP peer A can access EAP peer B with the credentials and information obtained from EAP authenticator C in the second phase. This access can be repeated without contacting the EAP authenticator, while the credentials given to A are still valid. The details of this phase are out of the scope of this document.

It is worth noting that the first phase with CoAP-EAP is required to join the EAP authenticator C's domain. Once it is performed with success, the communications are local to the EAP authenticator C's domain and there is no need to perform a new EAP authentication as long as the key material is still valid. When the keys are about to expire, the EAP peer can engage in a re-authentication as explained in Section 3.3, to renew the key material.

C.2. Example 2: Multi-domain with AAA infrastructures

We assume we have a device (A) of the domain acme.org, which uses a specific kind of credential (e.g., AKA) and intends to join the um.es domain. This user does not belong to this domain, for which first it performs a client registration using CoAP-EAP. For this, it interacts with the EAP authenticator's domain, which in turn communicates with an AAA infrastructure (acting as AAA client). Through the local AAA server communicate with the home AAA server to complete the authentication and integrate the device as a trustworthy entity into the domain of EAP authenticator C. In this scenario, the AS under the role of the EAP authenticator receives the key material from the AAA infrastructure

C.3. Example 3: Single domain with AAA infrastructure

As a University Campus, we have several Faculty buildings and each one has its criteria or policies in place to manage EAP peers under an AS. All buildings belong to the same domain (e.g., um.es). All these buildings are managed with AAA infrastructure. A new device (A) with credentials from the domain (e.g., um.es) will be able to perform the device registration with an EAP authenticator (C) of any building if they are managed by the same general domain.

C.4. Example 4: Single domain without AAA infrastructure

In another case, without a AAA infrastructure, we have an EAP authenticator that has co-located the EAP server, and using EAP standalone mode we can manage all the devices within the same domain locally. Client registration of an EAP peer (A) with Controller (C) can also be performed in the same manner.

C.5. Other use cases

C.5.1. CoAP-EAP for network access control

One of the first steps for an EAP peer's life cycle is to perform the authentication to gain access to the network. To do so, the device first must be authenticated and granted authorization to gain access to the network. Additionally, security parameters such as credentials can be derived from the authentication process allowing the trustworthy operation of the EAP peer in a particular network by joining the security domain. By using EAP, we can achieve this with flexibility and scalability, because of the different EAP methods available and the ability to rely on AAA infrastructures if needed to support multi-domain scenarios, which is a key feature when the EAP peers deployed under the same security domain, belong to different organizations.

Given that EAP is also used for network access control, we can adapt this service to other technologies. For instance, to provide network access control to very constrained technologies (e.g., LoRa network). Authors in [lo-coap-eap] provide a study of a minimal version of CoAP-EAP for LPWAN networks with interesting results. In this specific case, we could leverage the compression by SCHC for CoAP [RFC8824].

C.5.2. CoAP-EAP for service authentication

It is not uncommon that the infrastructure where the device is deployed and the services of the EAP peer are managed by different organizations. Therefore, in addition to the authentication for network access control, we have to consider the possibility of a secondary authentication to access different services. This process of authentication, for example, will provide the necessary key material to establish a secure channel and interact with the entity in charge of granting access to different services.

In 5G, for example, consider primary and secondary authentication using EAP [TS133.501].

Acknowledgments

We would like to thank the reviewers of this work: Paul Wouters, Heikki Vatiainen, Josh Howlett, Deb Cooley, Eliot Lear, Alan DeKok, Carsten Bormann, Mohit Sethi, Benjamin Kaduk, Christian Amsuss, John Mattsson, Goran Selander, Alexandre Petrescu, Pedro Moreno-Sanchez and Eduardo Ingles-Sanchez.

We would also like to thank Gabriel Lopez-Millan for the first review of this document, and we would like to thank Ivan Jimenez-Sanchez for the first proof-of-concept implementation of this idea, and Julian Niklas Schimmelpfennig for the implementation of the Erbium-based IoT device implementation.

And thank for their valuable comments to Alexander Pelov and Laurent Toutain, especially for the potential optimizations of CoAP-EAP.

Authors' Addresses

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
30100 Murcia
Spain
Email: rafa@um.es

Dan Garcia-Carrillo
University of Oviedo
Calle Luis Ortiz Berrocal S/N, Edificio Polivalente
33203 Gijon Asturias
Spain
Email: garciadan@uniovi.es

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2021

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

February 22, 2021

Group OSCORE Profile of the Authentication and Authorization for Constrained Environments Framework
draft-tiloca-ace-group-oscore-profile-05

Abstract

This document specifies a profile for the Authentication and Authorization for Constrained Environments (ACE) framework. The profile uses Group OSCORE to provide communication security between a Client and a (set of) Resource Server(s) as members of an OSCORE Group. The profile securely binds an OAuth 2.0 Access Token with the public key of the Client associated to the signing private key used in the OSCORE group. The profile uses Group OSCORE to achieve server authentication, as well as proof-of-possession for the Client's public key. Also, it provides proof of the Client's membership to the correct OSCORE group, by binding the Access Token to information from the Group OSCORE Security Context, thus allowing the Resource Server(s) to verify the Client's membership upon receiving a message protected with Group OSCORE from the Client.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 26, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Terminology 6
2. Protocol Overview 6
2.1. Pre-Conditions 8
2.2. Access Token Retrieval 9
2.3. Access Token Posting 9
2.4. Secure Communication 10
3. Client-AS Communication 10
3.1. C-to-AS: POST to Token Endpoint 10
3.1.1. 'context_id' Parameter 12
3.1.2. 'salt_input' Parameter 12
3.1.3. 'client_cred_verify' Parameter 13
3.2. AS-to-C: Access Token 13
3.2.1. Salt Input Claim 16
3.2.2. Context ID Input Claim 17
4. Client-RS Communication 17
4.1. C-to-RS POST to authz-info Endpoint 18
4.2. RS-to-C: 2.01 (Created) 18
4.3. Client-RS Secure Communication 19
4.3.1. Client Side 19
4.3.2. Resource Server Side 20
4.4. Access Rights Verification 20
4.5. Change of Client's Public Key in the Group 21
5. Secure Communication with the AS 21
6. Discarding the Security Context 22
7. CBOR Mappings 22
8. Security Considerations 23
9. Privacy Considerations 23
10. IANA Considerations 24
10.1. ACE Profile Registry 24
10.2. OAuth Parameters Registry 25

10.3. OAuth Parameters CBOR Mappings Registry 25
10.4. CBOR Web Token Claims Registry 26
10.5. TLS Exporter Label Registry 27
11. References 28
11.1. Normative References 28
11.2. Informative References 30
Appendix A. Dual Mode (Group OSCORE & OSCORE) 31
A.1. Protocol Overview 32
A.1.1. Pre-Conditions 34
A.1.2. Access Token Posting 34
A.1.3. Setup of the Pairwise OSCORE Security Context 34
A.1.4. Secure Communication 35
A.2. Client-AS Communication 36
A.2.1. C-to-AS: POST to Token Endpoint 36
A.2.2. AS-to-C: Access Token 39
A.3. Client-RS Communication 45
A.3.1. C-to-RS POST to authz-info Endpoint 46
A.3.2. RS-to-C: 2.01 (Created) 47
A.3.3. OSCORE Setup - Client Side 48
A.3.4. OSCORE Setup - Resource Server Side 51
A.3.5. Access Rights Verification 53
A.3.6. Change of Client's Public Key in the Group 53
A.4. Secure Communication with the AS 54
A.5. Discarding the Security Context 54
A.6. CBOR Mappings 55
A.7. Security Considerations 55
A.8. Privacy Considerations 55
Appendix B. Profile Requirements 56
Acknowledgments 57
Authors' Addresses 57

1. Introduction

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

For some of such applications, it may be just fine to enforce access control in a straightforward fashion. That is, any Client authorized to join the group, hence to get the group key material, can be also implicitly authorized to perform any action at any resource of any Server in the group. An example of application where such implicit authorization might be used is a simple lighting scenario, where the lightbulbs are the Servers, while the user account on an app on the user's phone is the Client. In this case, it might be fine to not

require additional authorization evidence from any user account, if it is acceptable that any current group member is also authorized to switch on and off any light, or to check their status.

However, in different instances of such applications, the approach above is not desirable, as different group members are intended to have different access rights to resources of other group members. That is, access control to the secure group communication channel and access control to the resource space provided by servers in the group should remain logically separated domains. For instance, a more fine-grained approach is required in the two following use cases.

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

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

In the cases above, being a legitimate group member and owning the group key material is not supposed to imply any particular access rights. Also, introducing a different security group for each different set of access rights would result in additional key material to distribute and manage. In particular, if the access rights for a single node change, this would require to evict that

node from the current group, followed by that node joining a different group aligned with its new access rights. Moreover, the key material of both groups would have to be renewed for their current members. Overall, this would have a non negligible impact on operations and performance in the system.

A fine-grained access control model can be rather enforced within a same group, by using the Authentication and Authorization for Constrained Environments (ACE) framework [I-D.ietf-ace-oauth-authz]. That is, a Client has to first obtain authorization credentials in the form of an Access Token, and post it to the Resource Server(s) in the group before accessing the intended resources.

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

This document specifies a profile of ACE, where a Client uses CoAP [RFC7252] or CoAP over IP multicast [I-D.ietf-core-groupcomm-bis] to communicate to one or multiple Resource Servers, which are members of an application group and share a common set of resources. This profile uses Group OSCORE [I-D.ietf-core-oscore-groupcomm] as the security protocol to protect messages exchanged between the Client and the Resource Servers. Hence, it requires that both the Client and the Resource Servers have previously joined the same OSCORE group.

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

The Client proves its access to be authorized to the Resource Server by using an Access Token, which is bound to a key (the proof-of-possession key). This profile uses Group OSCORE to achieve server authentication, as well as proof-of-possession for the Client's public key associated to the signing private key used in an OSCORE group. Note that the proof of possession is not done by a dedicated protocol element, but rather occurs after the first Group OSCORE exchange. Furthermore, this profile provides proof of the Client's membership to the correct OSCORE group, by binding the Access Token to the Client's public key and information from the pre-established Group OSCORE Security Context, thus allowing the Resource Server to

verify this upon reception of a messages protected with Group OSCORE from the Client.

OSCORE [RFC8613] specifies how to use COSE [I-D.ietf-cose-rfc8152bis-struct][I-D.ietf-cose-rfc8152bis-algs] to secure CoAP messages. Group OSCORE builds on OSCORE to provide secure group communication, and ensures source authentication either: by means of digital counter signatures embedded in protected messages (in group mode); by protecting messages with pairwise key material derived from the asymmetric keys of the two peers exchanging the message (in pairwise mode).

1.1. Terminology

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

Readers are expected to be familiar with the terms and concepts related to CBOR [RFC8949], COSE [I-D.ietf-cose-rfc8152bis-struct][I-D.ietf-cose-rfc8152bis-algs], CoAP [RFC7252], OSCORE [RFC8613] and Group OSCORE [I-D.ietf-core-oscore-groupcomm]. These include the concept of Group Manager, as the entity responsible for a set of groups where communications among members are secured with Group OSCORE.

Readers are expected to be familiar with the terms and concepts described in the ACE framework for authentication and authorization [I-D.ietf-ace-oauth-authz], as well as in the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile]. The terminology for entities in the considered architecture is defined in OAuth 2.0 [RFC6749]. In particular, this includes Client (C), Resource Server (RS), and Authorization Server (AS).

Note that, unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the AS, and /authz-info at the RS. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol".

2. Protocol Overview

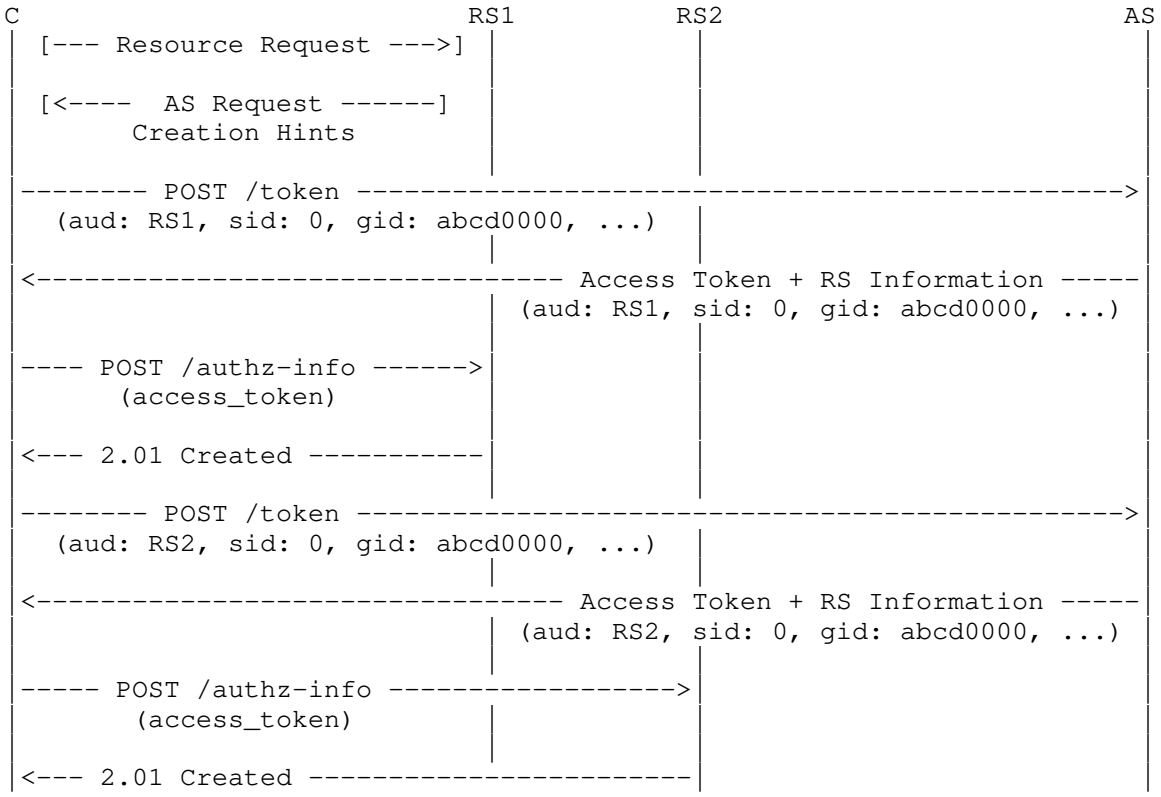
This section provides an overview of this profile, i.e. on how to use the ACE framework for authentication and authorization [I-D.ietf-ace-oauth-authz] to secure communications between a Client

and a (set of) Resource Server(s) using Group OSCORE [I-D.ietf-core-oscore-groupcomm].

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

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

An overview of the protocol flow for this profile is shown in Figure 1. In the figure, it is assumed that both RS1 and RS2 are associated with the same AS. It is also assumed that C, RS1 and RS2 have previously joined an OSCORE group with Group Identifier (gid) "abcd0000", and got assigned Sender ID (sid) "0", "1" and "2" in the group, respectively. The names of messages coincide with those of [I-D.ietf-ace-oauth-authz] when applicable.



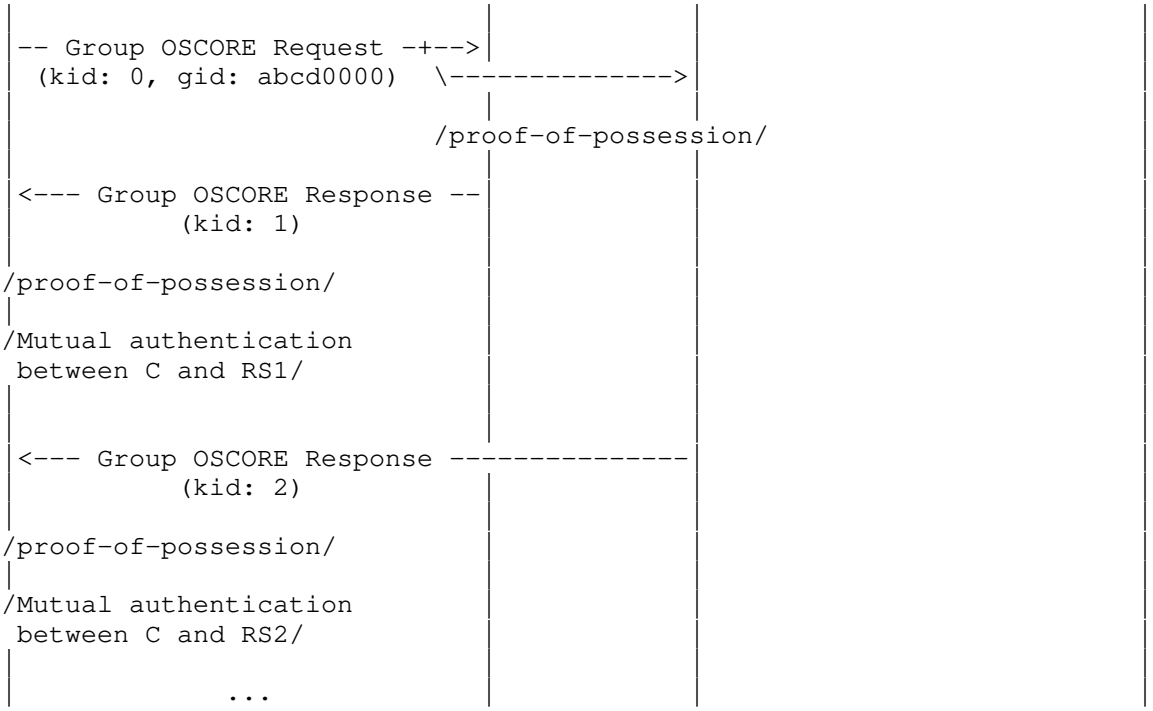


Figure 1: Protocol Overview.

2.1. Pre-Conditions

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

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

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

2.2. Access Token Retrieval

This profile requires that the Client retrieves an Access Token from the AS for the resource(s) it wants to access on each of the RSs, using the /token endpoint, as specified in Section 5.8 of [I-D.ietf-ace-oauth-authz]. In a general case, it can be assumed that different RSs are associated to different ASs, even if the RSs are members of a same OSCORE group.

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

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

The Access Token request and response MUST be confidentiality-protected and ensure authenticity. This profile RECOMMENDS the use of OSCORE between the Client and the AS, to reduce the number of libraries the client has to support. Other protocols fulfilling the security requirements defined in Section 5 of [I-D.ietf-ace-oauth-authz] (such as TLS [RFC8446] or DTLS [RFC6347][I-D.ietf-tls-dtls13]) MAY be used additionally or instead.

2.3. Access Token Posting

After having retrieved the Access Token from the AS, the Client posts the Access Token to the RS, using the /authz-info endpoint and mechanisms specified in Section 5.10 of [I-D.ietf-ace-oauth-authz], as well as Content-Format = application/ace+cbor. When using this profile, the communication with the /authz-info endpoint is not protected.

If the Access Token is valid, the RS replies to this POST request with a 2.01 (Created) response with Content-Format = application/ace+cbor. Also, the RS associates the received Access Token with the Group OSCORE Security Context identified by the Group Identifier specified in the Access Token, following Section 3.2 of [RFC8613]. In practice, the RS maintains a collection of Security Contexts with associated authorization information, for all the clients that it is currently communicating with, and the authorization information is a policy used as input when processing requests from those clients.

Finally, the RS stores the association between i) the authorization information from the Access Token; and ii) the Group Identifier of the OSCORE group together with the Sender ID and the public key of the Client in that group. This binds the Access Token with the Group OSCORE Security Context of the OSCORE group.

Finally, when the Client communicates with the RS using the Group OSCORE Security Context, the RS verifies that the Client is a legitimate member of the OSCORE group and especially the exact group member with the same Sender ID associated to the Access Token. This occurs when verifying a request protected with Group OSCORE, since it embeds a counter signature computed also over the Client's Sender ID included in the message.

2.4. Secure Communication

The Client can send a request protected with Group OSCORE [I-D.ietf-core-oscore-groupcomm] to the RS. This can be a unicast request addressed to the RS, or a multicast request addressed to the OSCORE group where the RS is also a member. To this end, the Client uses the Group OSCORE Security Context already established upon joining the OSCORE group, e.g. by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore]. The RS may send a response back to the Client, protecting it by means of the same Group OSCORE Security Context.

3. Client-AS Communication

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

The Access Token MUST be bound to the public key of the client as proof-of-possession key (pop-key), by means of the 'cnf' claim.

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

The Client-to-AS request is specified in Section 5.8.1 of [I-D.ietf-ace-oauth-authz]. The Client MUST send this POST request to the /token endpoint over a secure channel that guarantees authentication, message integrity and confidentiality.

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

- 'context_id', defined in Section 3.1.1 of this specification. This parameter specifies the Group Identifier (GID), i.e. the Id Context of an OSCORE group where the Client and the RS are currently members. In particular, the Client wishes to communicate with the RS using the Group OSCORE Security Context associated to that OSCORE group.

- 'salt_input', defined in Section 3.1.2 of this specification. This parameter includes the Sender ID that the Client has in the OSCORE group whose GID is specified in the 'context_id' parameter above.

- 'req_cnf', defined in Section 3.1 of [I-D.ietf-ace-oauth-params]. This parameter includes the public key associated to the signing private key that the Client uses in the OSCORE group whose GID is specified in the 'context_id' parameter above. This public key will be used as the pop-key bound to the Access Token.

- 'client_cred_verify', defined in Section 3.1.3 of this specification. This parameter includes a signature computed by the Client, by using the private key associated to the public key in the 'req_cnf' parameter above. This allows the AS to verify that the Client indeed owns the private key associated to that public key, as its alleged identity credential within the OSCORE group. The information to be signed MUST be the byte representation of a quantity that uniquely represents the secure communication association between the Client and the AS. It is RECOMMENDED that the Client considers the following as information to sign.

  * If the Client and the AS communicate over (D)TLS, the information to sign is an exporter value computed as defined in Section 7.5 of [RFC8446]. In particular, the exporter label MUST be 'EXPORTER-ACE-Sign-Challenge-Client-AS' defined in Section 10.5 of this specification, together with an empty 'context_value', and 32 bytes as 'key_length'.

  * If the Client and the AS communicate over OSCORE, the information to sign is the output PRK of a HKDF-Extract step [RFC5869], i.e. PRK = HMAC-Hash(salt, IKM). In particular, 'salt' takes (x1 | x2), where x1 is the ID Context of the OSCORE Security Context between the Client and the AS, x2 is the Sender ID of the Client in that Context, and | denotes byte string concatenation. Also, 'IKM' is the OSCORE Master Secret of the OSCORE Security Context between the Client and the AS. The HKDF MUST be one of the HMAC-based HKDF [RFC5869] algorithms defined for COSE [I-D.ietf-cose-rfc8152bis-algs]. HKDF SHA-256 is mandatory to implement.

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

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

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

3.1.1. 'context_id' Parameter

The 'context_id' parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.8.1. of [I-D.ietf-ace-oauth-authz]. This parameter provides a value that the Client wishes to use with the RS as a hint for a security context. Its exact content is profile specific.

3.1.2. 'salt_input' Parameter

The 'salt_input' parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.8.1. of [I-D.ietf-ace-oauth-authz]. This parameter provides a value that the Client wishes to use as part of a salt with the RS, for deriving cryptographic key material. Its exact content is profile specific.

3.1.3. 'client_cred_verify' Parameter

The 'client_cred_verify' parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.8.1. of [I-D.ietf-ace-oauth-authz]. This parameter provides a signature computed by the Client to prove the possession of its own private key.

3.2. AS-to-C: Access Token

After having verified the POST request to the /token endpoint and that the Client is authorized to obtain an Access Token corresponding to its Access Token request, the AS MUST verify the signature in the 'client_cred_verify' parameter, by using the public key specified in the 'req_cnf' parameter. If the verification fails, the AS considers the Client request invalid.

If all verifications are successful, the AS responds as defined in Section 5.8.2 of [I-D.ietf-ace-oauth-authz]. If the Client request was invalid, or not authorized, the AS returns an error response as described in Section 5.8.3 of [I-D.ietf-ace-oauth-authz].

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

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

- The same parameter 'profile' included in the Token Response to the Client.

- The salt input specified in the 'salt_input' parameter of the Token Request. If the Access Token is a CWT, the content of the 'salt_input' parameter MUST be placed in the 'salt_input' claim of the Access Token, defined in Section 3.2.1 of this specification.

- The Context Id input specified in the 'context_id' parameter of the Token Request. If the Access Token is a CWT, the content of the 'context_id' parameter MUST be placed in the 'contextId_input' claim of the Access Token, defined in Section 3.2.2 of this specification.

- The public key that the client uses in the OSCORE group and specified in the 'req_cnf' parameter of the Token request. If the Access Token is a CWT, the public key MUST be specified in the 'cnf' claim, which follows the syntax from Section 3.1 of [RFC8747] when including Value Type "COSE_Key" (1) and specifying an asymmetric key. Alternative Value Types defined in future specifications are fine to consider if indicating a non-encrypted asymmetric key.

Figure 3 shows an example of such an AS response, with payload in CBOR diagnostic notation without the tag and value abbreviations.

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

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

Figure 4 shows an example CWT Claims Set, containing the Client's public key in the group (as pop-key) in the 'cnf' claim, in CBOR diagnostic notation without tag and value abbreviations.

```
{
  "aud" : "tempSensorInLivingRoom",
  "iat" : "1360189224",
  "exp" : "1360289224",
  "scope" :  "temperature_g firmware_p",
  "cnf" : {
    "COSE_Key" : {
      "kty" : EC2,
      "crv" : P-256,
      "x" : h'd7cc072de2205bdc1537a543d53c60a6acb62eccd890c7fa
             27c9e354089bbe13',
      "y" : h'f95e1d4b851a2cc80fff87d8e23f22afb725d535e515d020
             731e79a3b4e47120'
  },
  "salt_input" : h'00',
  "contextId_input" : h'abcd0000'
}
```

Figure 4: Example CWT Claims Set with OSCORE parameters (CBOR diagnostic notation).

The same CWT Claims Set as in Figure 4 and encoded in CBOR is shown in Figure 5, using the value abbreviations defined in [I-D.ietf-ace-oauth-authz] and [RFC8747]. The bytes in hexadecimal are reported in the first column, while their corresponding CBOR meaning is reported after the '#' sign on the second column, for easiness of readability.

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

```
A7                                      # map(7)
   03                                   # unsigned(3)
   76                                   # text(22)
      74656D7053656E736F72496E4C6976696E67526F6F6D
   06                                   # unsigned(6)
   1A 5112D728                          # unsigned(1360189224)
   04                                   # unsigned(4)
   1A 51145DC8                          # unsigned(1360289224)
   09                                   # unsigned(9)
   78 18                                # text(24)
      74656D7065726174572655F67206669726D776172655F70
   08                                   # unsigned(8)
   A1                                   # map(1)
      01                                # unsigned(1)
      A4                                # map(4)
         01                             # unsigned(1)
         02                             # unsigned(2)
         20                             # negative(0)
         01                             # unsigned(1)
         21                             # negative(1)
         58 20                          # bytes(32)
            D7CC072DE2205BDC1537A543D53C60A6ACB62ECCD890C7FA27C9
            E354089BBE13
         22                             # negative(2)
         58 20                          # bytes(32)
            F95E1D4B851A2CC80FFF87D8E23F22AFB725D535E515D020731E
            79A3B4E47120
   18 3C                                # unsigned(60)
   41                                   # bytes(1)
      00
   18 3D                                # unsigned(61)
   44                                   # bytes(4)
      ABCD0000
```

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

3.2.1. Salt Input Claim

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

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

3.2.2. Context ID Input Claim

The 'contextId_input' claim provides a value that the Client requesting the Access Token wishes to use with the RS, as a hint for a security context.

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

4. Client-RS Communication

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

The proof-of-possession required to bind the Access Token to the Client is explicitly performed when the RS receives and verifies a request from the Client protected with Group OSCORE, either with the group mode (see Section 8 of [I-D.ietf-core-oscore-groupcomm]) or with the pairwise mode (see Section 9 of [I-D.ietf-core-oscore-groupcomm]).

In particular, the RS uses the Client's public key bound to the Access Token, either when verifying the counter signature of the request (if protected with the group mode), or when verifying the request as integrity-protected with pairwise key material derived from the two peers' asymmetric keys (if protected with the pairwise mode). In either case, the RS also authenticates the Client.

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

Therefore, an attacker using a stolen Access Token cannot generate a valid Group OSCORE message signed with the Client's private key, and thus cannot prove possession of the pop-key bound to the Access Token. Also, if a Client legitimately owns an Access Token but has not joined the OSCORE group, it cannot generate a valid Group OSCORE message, as it does not own the necessary key material shared among the group members.

Furthermore, a Client C1 is supposed to obtain a valid Access Token from the AS, as including the public key associated to its own

signing key used in the OSCORE group, together with its own Sender ID in that OSCORE group (see Section 3.1). This makes it possible for the RS receiving an Access Token to verify with the Group Manager of that OSCORE group whether such a Client has indeed that Sender ID and that public key in the OSCORE group.

As a consequence, a different Client C2, also member of the same OSCORE group, is not able to impersonate C1, by: i) getting a valid Access Token, specifying the Sender ID of C1 and a different (made-up) public key; ii) successfully posting the Access Token to RS; and then iii) attempting to communicate using Group OSCORE impersonating C1, while blaming C1 for the consequences.

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

The Client posts the Access Token to the /authz-info endpoint of the RS, as defined in Section 5.10.1 of [I-D.ietf-ace-oauth-authz].

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

The RS MUST verify the validity of the Access Token as defined in Section 5.10.1 of [I-D.ietf-ace-oauth-authz], with the following additions.

- The RS checks that the claims 'salt_input', 'contextId_input' and 'cnf' are included in the Access Token.

- The RS considers the content of the 'cnf' claim as the public key associated to the signing private key of the Client in the OSCORE group, whose GID is specified in the 'contextId_input' claim.

  If it does not already store that public key, the RS MUST request it to the Group Manager of the OSCORE group as described in [I-D.ietf-ace-key-groupcomm-oscore], specifying the Sender ID of that Client in the OSCORE group, i.e. the value of the 'salt_input' claim above. The RS MUST check that the key retrieved from the Group Manager matches the one retrieved from the 'cnf' claim. When doing so, the 'kid' parameter of the COSE_Key, if present, MUST NOT be considered for the comparison.

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

If the Access Token is valid and further checks on its content are successful, the RS associates the authorization information from the Access Token with the Group OSCORE Security Context.

In particular, the RS associates the authorization information from the Access Token with the 3-tuple (GID, SaltInput, PubKey), where GID is the Group Identifier of the OSCORE Group, while SaltInput and PubKey are the Sender ID and the public key that the Client uses in that OSCORE group, respectively. These can be retrieved from the 'contextId_input', 'salt_input' and 'cnf' claims of the Access Token, respectively.

The RS MUST keep this association up-to-date over time, as the 3-tuple (GID, SaltInput, PubKey) associated to the Access Token might change. In particular:

- If the OSCORE group is rekeyed (see Section 3.1 of [I-D.ietf-core-oscore-groupcomm] and Section 18 of [I-D.ietf-ace-key-groupcomm-oscore]), the Group Identifier also changes in the group, and the new one replaces the current 'GID' value in the 3-tuple.

- If the Client requests and obtains a new OSCORE Sender ID from the Group Manager (see Section 3.1 of [I-D.ietf-core-oscore-groupcomm] and Section 9 of [I-D.ietf-ace-key-groupcomm-oscore]), the new Sender ID replaces the current 'SaltInput' value in the 3-tuple.

Finally, the RS MUST send a 2.01 (Created) response to the Client, as defined in Section 5.10.1 of [I-D.ietf-ace-oauth-authz].

4.3. Client-RS Secure Communication

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

4.3.1. Client Side

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

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

4.3.2. Resource Server Side

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

When processing an incoming request protected with Group OSCORE, the RS MUST consider as valid public key of the Client only the public key specified in the stored Access Token. As defined in Section 4.5, a possible change of public key requires the Client to upload to the RS a new Access Token bound to the new public key.

Additionally, for every incoming request, if Group OSCORE verification succeeds, the verification of access rights is performed as described in Section 4.4.

After the expiration of the Access Token related to a Group OSCORE Security Context, if the Client uses the Group OSCORE Security Context to send a request for any resource intended for OSCORE group members and that requires an active Access Token, the RS MUST respond with a 4.01 (Unauthorized) error message protected with the Group OSCORE Security Context.

4.4. Access Rights Verification

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

If the Group OSCORE verification succeeds, and the target resource requires authorization, the RS retrieves the authorization information from the Access Token associated to the Group OSCORE Security Context. Then, the RS MUST verify that the action requested on the resource is authorized.

The response code MUST be 4.01 (Unauthorized) if the RS has no valid Access Token for the Client. If the RS has an Access Token for the Client but no actions are authorized on the target resource, the RS MUST reject the request with a 4.03 (Forbidden). If the RS has an Access Token for the Client but the requested action is not authorized, the RS MUST reject the request with a 4.05 (Method Not Allowed).

4.5. Change of Client's Public Key in the Group

During its membership in the OSCORE group, the client might change the public key it uses in the group. When this happens, the Client uploads the new public key to the Group Manager, as defined in Section 11 of [I-D.ietf-ace-key-groupcomm-oscore].

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

1. The Client requests a new Access Token to the AS, as defined in Section 3. In particular, when sending the POST request as defined in Section 3.1, the Client indicates:

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

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

   * The new public key it uses in the OSCORE group, as value of the 'req_cnf' parameter.

   * The proof-of-possession signature corresponding to the new public key, as value of the 'client_cred_verify' parameter.

2. After receiving the response from the AS (see Section 3.2), the Client performs the same exchanges with the RS as defined in Section 4.

When receiving the new Access Token, the RS performs the same steps defined in Section 4.2, with the following addition, in case the new Access Token is successfully verified and stored. The RS also deletes the old Access Token, i.e. the one whose associated 3-tuple has the same GID and SaltInput values as in the 3-tuple including the new public key of the Client and associated to the new Access Token.

5. Secure Communication with the AS

As specified in the ACE framework (Sections 5.8 and 5.9 of [I-D.ietf-ace-oauth-authz]), the requesting entity (RS and/or Client) and the AS communicate via the /introspection or /token endpoint. The use of CoAP and OSCORE [RFC8613] for this communication is RECOMMENDED in this profile. Other protocols fulfilling the security requirements defined in Section 5 of [I-D.ietf-ace-oauth-authz] (such as HTTP and DTLS or TLS) MAY be used instead.

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

6. Discarding the Security Context

As members of an OSCORE group, the Client and the RS may independently leave the group or be forced to, e.g. if compromised or suspected so. Upon leaving the OSCORE group, the Client or RS also discards the Group OSCORE Security Context, which may anyway be renewed by the Group Manager through a group rekeying process (see Section 3.1 of [I-D.ietf-core-oscore-groupcomm]).

The Client or RS can acquire a new Group OSCORE Security Context, by re-joining the OSCORE group, e.g. by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore]. In such a case, the Client SHOULD request a new Access Token and post it to the RS.

7. CBOR Mappings

The new parameters defined in this document MUST be mapped to CBOR types as specified in Figure 6, using the given integer abbreviation for the map key.

Parameter name	CBOR Key	Value Type
context_id	TBD1	bstr
salt_input	TBD2	bstr
client_cred_verify	TBD3	bstr

Figure 6: CBOR mappings for new parameters.

The new claims defined in this document MUST be mapped to CBOR types as specified in Figure 7, using the given integer abbreviation for the map key.

Claim name	CBOR Key	Value Type
salt_input	TBD4	bstr
contextId_input	TBD5	bstr

Figure 7: CBOR mappings for new claims.

8. Security Considerations

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

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

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

Group OSCORE ensures source authentication of messages both in group mode (see Section 8 of [I-D.ietf-core-oscore-groupcomm]) and in pairwise mode (see Section 9 of [I-D.ietf-core-oscore-groupcomm]).

When protecting an outgoing message in group mode, the sender uses its private key to compute a digital counter signature, which is embedded in the protected message. The group mode can be used to protect messages sent over multicast to multiple recipients, or sent over unicast to one recipient.

When protecting an outgoing message in pairwise mode, the sender uses a pairwise symmetric key, as derived from the asymmetric keys of the two peers exchanging the message. The pairwise mode can be used to protect only messages sent over unicast to one recipient.

9. Privacy Considerations

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

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

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

Although it may be encrypted, the Access Token is sent in the clear to the /authz-info endpoint at the RS. Thus, if the Client uses the same single Access Token from multiple locations with multiple Resource Servers, it can risk being tracked through the Access Token's value.

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

10. IANA Considerations

This document has the following actions for IANA.

10.1. ACE Profile Registry

IANA is asked to enter the following value into the "ACE Profile" Registry defined in Section 8.8 of [I-D.ietf-ace-oauth-authz].

- Name: coap_group_oscore
- Description: Profile to secure communications between constrained nodes using the Authentication and Authorization for Constrained Environments framework, by enabling authentication and fine-grained authorization of members of an OSCORE group, that use a pre-established Group OSCORE Security Context to communicate with Group OSCORE. Optionally, the dual mode defined in Appendix A additionally establishes a pairwise OSCORE Security Context, and thus also enables OSCORE communication between two members of the OSCORE group.
- CBOR Value: TBD (value between 1 and 255)
- Reference: [[this document]]

10.2. OAuth Parameters Registry

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

- Name: "context_id"
- Parameter Usage Location: token request
- Change Controller: IESG
- Specification Document(s): Section 3.1.1 of [[this document]]

- Name: "salt_input"
- Parameter Usage Location: token request
- Change Controller: IESG
- Specification Document(s): Section 3.1.2 of [[this document]]

- Name: "client_cred_verify"
- Parameter Usage Location: token request
- Change Controller: IESG
- Specification Document(s): Section 3.1.3 of [[this document]]

- Name: "client_cred"
- Parameter Usage Location: token request
- Change Controller: IESG
- Specification Document(s): Appendix A.2.1.1 of [[this document]]

10.3. OAuth Parameters CBOR Mappings Registry

IANA is asked to enter the following values into the "OAuth Parameters CBOR Mappings" Registry defined in Section 8.10 of [I-D.ietf-ace-oauth-authz].

- Name: "context_id"

- CBOR Key: TBD1
- Value Type: bstr
- Reference: Section 3.1.1 of [[this document]]

- Name: "salt_input"
- CBOR Key: TBD2
- Value Type: bstr
- Reference: Section 3.1.2 of [[this document]]

- Name: "client_cred_verify"
- CBOR Key: TBD3
- Value Type: bstr
- Reference: Section 3.1.3 of [[this document]]

- Name: "client_cred"
- CBOR Key: TBD6
- Value Type: bstr
- Reference: Appendix A.2.1.1 of [[this document]]

10.4. CBOR Web Token Claims Registry

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

- Claim Name: "salt_input"
- Claim Description: Client provided salt input
- JWT Claim Name: "N/A"
- Claim Key: TBD4
- Claim Value Type(s): bstr

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

- Claim Name: "contextId_input"
- Claim Description: Client context id input
- JWT Claim Name: "N/A"
- Claim Key: TBD5
- Claim Value Type(s): bstr
- Change Controller: IESG
- Specification Document(s): Section 3.2.2 of [[this document]]

- Claim Name: "client_cred"
- Claim Description: Client Credential
- JWT Claim Name: "N/A"
- Claim Key: TBD7
- Claim Value Type(s): map
- Change Controller: IESG
- Specification Document(s): Appendix A.2.2.2 of [[this document]]

10.5. TLS Exporter Label Registry

IANA is asked to register the following entry in the "TLS Exporter Label" Registry defined in Section 6 of [RFC5705] and updated in Section 12 of [RFC8447].

- Value: EXPORTER-ACE-Sign-Challenge-Client-AS
- DTLS-OK: Y
- Recommended: N
- Reference: [[this document]] (Section 3.1)

11. References

11.1. Normative References

[I-D.ietf-ace-key-groupcomm-oscore]
Tiloca, M., Park, J., and F. Palombini, "Key Management for OSCORE Groups in ACE", draft-ietf-ace-key-groupcomm-oscore-10 (work in progress), February 2021.

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-37 (work in progress), February 2021.

[I-D.ietf-ace-oauth-params]
Seitz, L., "Additional OAuth Parameters for Authorization in Constrained Environments (ACE)", draft-ietf-ace-oauth-params-13 (work in progress), April 2020.

[I-D.ietf-ace-oscore-profile]
Palombini, F., Seitz, L., Selander, G., and M. Gunnarsson, "OSCORE Profile of the Authentication and Authorization for Constrained Environments Framework", draft-ietf-ace-oscore-profile-16 (work in progress), January 2021.

[I-D.ietf-core-groupcomm-bis]
Dijk, E., Wang, C., and M. Tiloca, "Group Communication for the Constrained Application Protocol (CoAP)", draft-ietf-core-groupcomm-bis-03 (work in progress), February 2021.

[I-D.ietf-core-oscore-groupcomm]
Tiloca, M., Selander, G., Palombini, F., Mattsson, J., and J. Park, "Group OSCORE - Secure Group Communication for CoAP", draft-ietf-core-oscore-groupcomm-11 (work in progress), February 2021.

[I-D.ietf-cose-rfc8152bis-algs]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", draft-ietf-cose-rfc8152bis-algs-12 (work in progress), September 2020.

[I-D.ietf-cose-rfc8152bis-struct]
Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", draft-ietf-cose-rfc8152bis-struct-15 (work in progress), February 2021.

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

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

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

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

[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/info/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/info/rfc7252>.

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

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

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/info/rfc8447>.

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

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

[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/info/rfc8949>.

11.2. Informative References

[I-D.ietf-ace-dtls-authorize]
Gerdes, S., Bergmann, O., Bormann, C., Selander, G., and L. Seitz, "Datagram Transport Layer Security (DTLS) Profile for Authentication and Authorization for Constrained Environments (ACE)", draft-ietf-ace-dtls-authorize-15 (work in progress), January 2021.

[I-D.ietf-ace-mqtt-tls-profile]
Sengul, C. and A. Kirby, "Message Queuing Telemetry Transport (MQTT)-TLS profile of Authentication and Authorization for Constrained Environments (ACE) Framework", draft-ietf-ace-mqtt-tls-profile-10 (work in progress), December 2020.

[I-D.ietf-tls-dtls13]
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", draft-ietf-tls-dtls13-41 (work in progress), February 2021.

[I-D.tiloca-core-oscore-discovery]
Tiloca, M., Amsuess, C., and P. Stok, "Discovery of OSCORE Groups with the CoRE Resource Directory", draft-tiloca-core-oscore-discovery-08 (work in progress), February 2021.

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

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

Appendix A. Dual Mode (Group OSCORE & OSCORE)

This appendix defines the dual mode of this profile, which allows using both OSCORE [RFC8613] and Group OSCORE [I-D.ietf-core-oscore-groupcomm] as security protocols, by still relying on a single Access Token.

That is, the dual mode of this profile specifies how a Client uses CoAP [RFC7252] to communicate to a single Resource Server, or CoAP over IP multicast [I-D.ietf-core-groupcomm-bis] to communicate to multiple Resource Servers that are members of a group and share a common set of resources.

In particular, the dual mode of this profile uses two complementary security protocols to provide secure communication between the Client and the Resource Server(s). That is, it defines the use of either OSCORE or Group OSCORE to protect unicast requests addressed to a single Resource Server, as well as possible responses. Additionally, it defines the use of Group OSCORE to protect multicast requests sent to a group of Resource Servers, as well as possible individual responses. Like in the main mode of this profile, the Client and the Resource Servers need to have already joined the same OSCORE group, for instance by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore], which is also based on ACE.

The Client proves its access to be authorized to the Resource Server by using an Access Token, which is bound to a key (the proof-of-possession key). This profile mode uses OSCORE to achieve proof of possession, and OSCORE or Group OSCORE to achieve server authentication.

Unlike in the main mode of this profile, where a public key is used as pop-key, this dual mode uses OSCORE-related, symmetric key material as pop-key instead. Furthermore, this dual mode provides proof of Client's membership to the correct OSCORE group, by securely binding the pre-established Group OSCORE Security Context to the pairwise OSCORE Security Context newly established between the Client and the Resource Server.

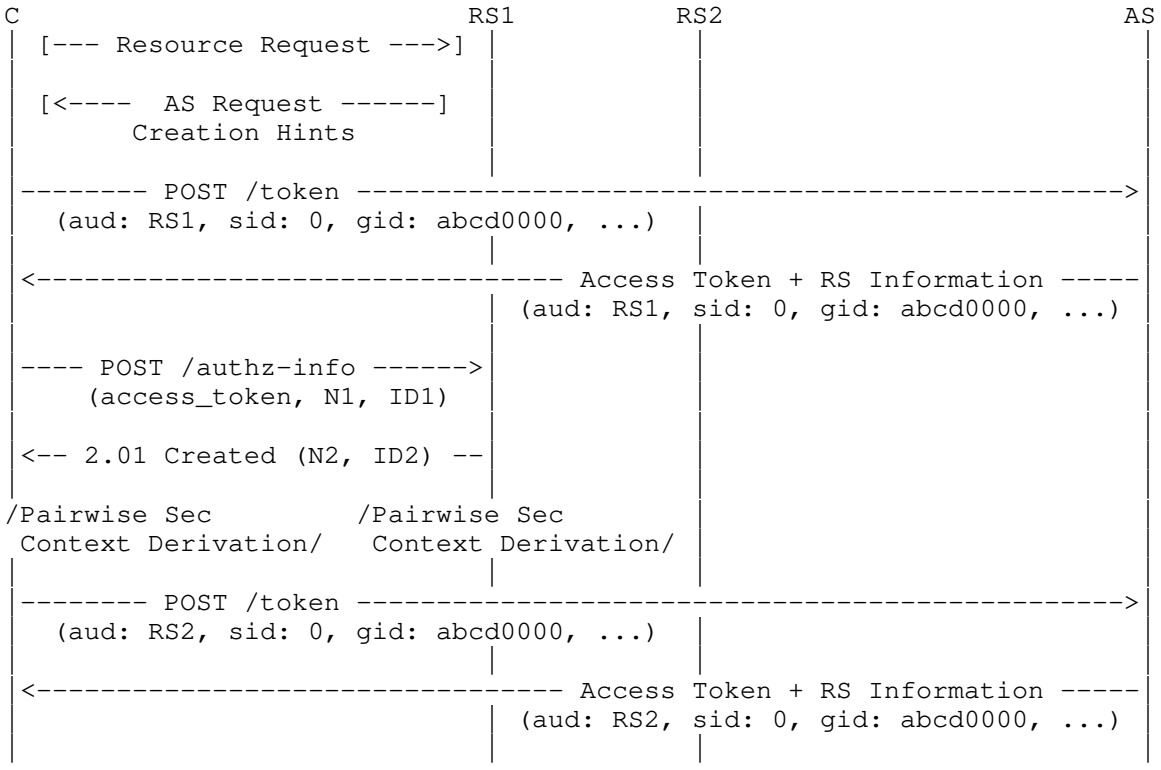
In addition to the terminology used for the main mode of this profile, the rest of this appendix refers also to "pairwise OSCORE Security Context" as to an OSCORE Security Context established between only one Client and one Resource Server, and used to communicate with OSCORE [RFC8613].

A.1. Protocol Overview

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

Just as for main mode of this profile overviewed in Section 2, the process for joining the OSCORE group through the respective Group Manager as defined in [I-D.ietf-ace-key-groupcomm-oscore] must take place before the process described in the rest of this section, and is out of the scope of this profile.

An overview of the protocol flow for the dual mode of this profile is shown in Figure 8. In the figure, it is assumed that both RS1 and RS2 are associated with the same AS. It is also assumed that C, RS1 and RS2 have previously joined an OSCORE group with Group Identifier (gid) "abcd0000", and got assigned Sender ID (sid) "0", "1" and "2" in the group, respectively. The names of messages coincide with those of [I-D.ietf-ace-oauth-authz] when applicable.



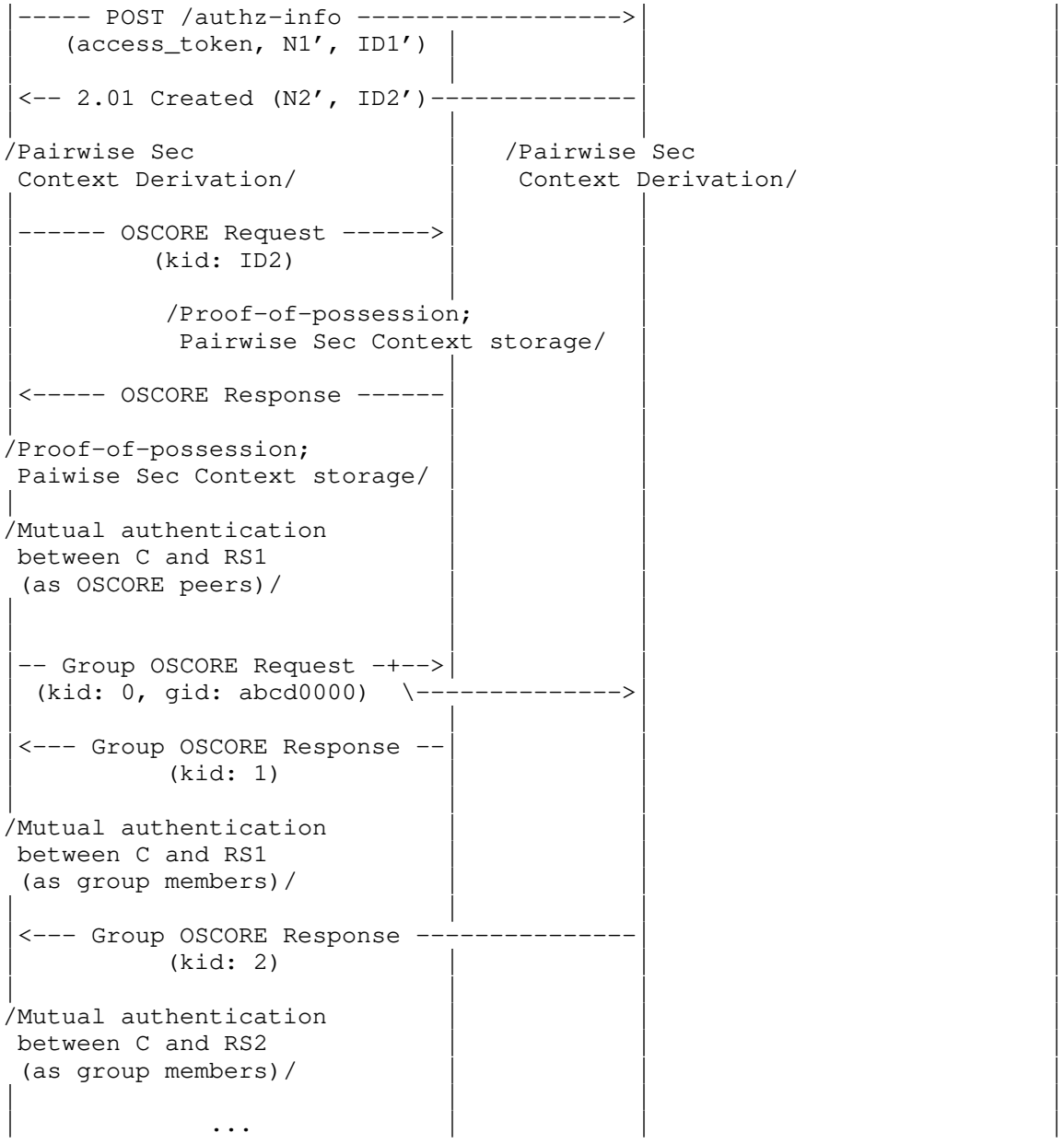


Figure 8: Protocol Overview.

A.1.1. Pre-Conditions

The same pre-conditions for the main mode of this profile (see Section 2.1) hold for the dual mode described in this appendix.

A.1.2. Access Token Posting

After having retrieved the Access Token from the AS, the Client generates a nonce N1 and an identifier ID1 unique in the sets of its own Recipient IDs from its pairwise OSCORE Security Contexts. The client then posts both the Access Token, N1 and its chosen ID to the RS, using the /authz-info endpoint and mechanisms specified in Section 5.10 of [I-D.ietf-ace-oauth-authz] and Content-Format = application/ace+cbor. When using the dual mode of this profile, the communication with the authz-info endpoint is not protected, except for update of access rights. Note that, when using the dual mode, this request can alternatively be protected with Group OSCORE, using the Group OSCORE Security Context paired with the pairwise OSCORE Security Context originally established with the first Access Token posting.

If the Access Token is valid, the RS replies to this POST request with a 2.01 (Created) response with Content-Format = application/ ace+cbor, which in a CBOR map contains a nonce N2 and an identifier ID2 unique in the sets of its own Recipient IDs from its pairwise OSCORE Security Contexts.

A.1.3. Setup of the Pairwise OSCORE Security Context

After sending the 2.01 (Created) response, the RS sets the ID Context of the pairwise OSCORE Security Context (see Section 3 of [RFC8613]) to the Group Identifier of the OSCORE group specified in the Access Token, concatenated with N1, concatenated with N2, concatenated with the value in the contextId parameter of the OSCORE_Input_Material provided in the 'cnf' claim of the Access Token.

Then, the RS derives the complete pairwise OSCORE Security Context associated with the received Access Token, following Section 3.2 of [RFC8613]. In practice, the RS maintains a collection of Security Contexts with associated authorization information, for all the clients that it is currently communicating with, and the authorization information is a policy used as input when processing requests from those clients.

During the derivation process, the RS uses: the ID Context above; the nonces N1 and N2; the identifier ID1 received from the Client, set as its own OSCORE Sender ID; the identifier ID2 provided to the Client, set as its Recipient ID for the Client; and the parameters in the

Access Token. The derivation process uses also the Master Secret of the OSCORE group, that the RS knows as a group member, as well as the Sender ID of the Client in the OSCORE group, which is specified in the Access Token. This ensures that the pairwise OSCORE Security Context is securely bound to the Group OSCORE Security Context of the OSCORE group.

Finally, the RS stores the association between i) the authorization information from the Access Token; and ii) the Group Identifier of the OSCORE group together with the Sender ID and the public key of the Client in that group.

After having received the nonce N2, the Client sets the ID Context in its pairwise OSCORE Security Context (see Section 3 of [RFC8613]) to the Group Identifier of the OSCORE group, concatenated with N1, concatenated with N2, concatenated with the value in the contextId parameter of the OSCORE_Input_Material provided in the 'cnf' parameter of the Access Token response from the AS. Then, the Client derives the complete pairwise OSCORE Security Context, following Section 3.2 of [RFC8613].

During the derivation process, the Client uses: the ID Context above, the nonces N1 and N2; the identifier ID1 provided to the RS, set as its own Recipient ID for the RS; the identifier ID2 received from the RS, set as its own OSCORE Sender ID; and the parameters received from the AS. The derivation process uses also the Master Secret of the OSCORE group, that the Client knows as a group member, as well as its own Sender ID in the OSCORE group.

When the Client communicates with the RS using the pairwise OSCORE Security Context, the RS achieves proof-of-possession of the credentials bound to the Access Token. Also, the RS verifies that the Client is a legitimate member of the OSCORE group.

A.1.4. Secure Communication

Other than starting the communication with the RS using Group OSCORE as described in Section 4.3, the Client can send to the RS a request protected with OSCORE, using the pairwise OSCORE Security Context.

If the request is successfully verified, then the RS stores the pairwise OSCORE Security Context, and uses it to protect the possible response, as well as further communications with the Client, until the Access Token is deleted, due to e.g. expiration. This pairwise OSCORE Security Context is discarded when an Access Token (whether the same or different) is used to successfully derive a new pairwise OSCORE Security Context.

As discussed in Section 7 of [I-D.ietf-ace-oscore-profile], the use of random nonces N1 and N2 during the exchange between the Client and the RS prevents the reuse of AEAD nonces and keys with different messages. Reuse might otherwise occur when Client and RS derive a new pairwise OSCORE Security Context from an existing (non-expired) Access Token, e.g. in case of reboot of either the Client or the RS, and might lead to loss of both confidentiality and integrity.

Additionally, just as per the main mode of this profile (see Section 4.3), the Client and RS can also securely communicate by protecting messages with Group OSCORE, using the Group OSCORE Security Context already established upon joining the OSCORE group.

A.2. Client-AS Communication

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

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

The proof-of-possession key (pop-key) received from the AS in this exchange MUST be used to build the Master Secret in OSCORE (see Appendix A.3.3 and Appendix A.3.4).

A.2.1. C-to-AS: POST to Token Endpoint

The Client-to-AS request is specified in Section 5.8.1 of [I-D.ietf-ace-oauth-authz]. The Client MUST send this POST request to the /token endpoint over a secure channel that guarantees authentication, message integrity and confidentiality.

The POST request is formatted as the analogous Client-to-AS request in the main mode of this profile (see Section 3.1), with the following modifications.

- The parameter 'req_cnf' MUST NOT be included in the payload.
- The parameter 'client_cred', defined in Appendix A.2.1.1 of this specification, MUST be included in the payload. This parameter includes the public key associated to the signing private key that the Client uses in the OSCORE group, whose identifier is indicated in the 'context_id' parameter.

- The signature included in the parameter 'client_cred_verify' is computed by using the private key associated to the public key in the 'client_cred' parameter above.

An example of such a request, with payload in CBOR diagnostic notation without the tag and value abbreviations is reported in Figure 9.

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

Figure 9: Example C-to-AS POST /token request for an Access Token bound to a symmetric key.

Later on, the Client may want to update its current access rights, without changing the existing pairwise OSCORE Security Context with the RS. In this case, the Client MUST include in its POST request to the /token endpoint a 'req_cnf' parameter, defined in Section 3.1 of [I-D.ietf-ace-oauth-params], which MUST include a 'kid' field, as defined in Section 3.1 of [RFC8747]. The 'kid' field has as value a CBOR byte string containing the OSCORE_Input_Material Identifier (assigned as discussed in Appendix A.2.2).

This identifier, together with other information such as audience, can be used by the AS to determine the shared secret bound to the proof-of-possession Access Token and therefore MUST identify a symmetric key that was previously generated by the AS as a shared

secret for the communication between the Client and the RS. The AS MUST verify that the received value identifies a proof-of-possession key that has previously been issued to the requesting Client. If that is not the case, the Client-to-AS request MUST be declined with the error code 'invalid_request' as defined in Section 5.8.3 of [I-D.ietf-ace-oauth-authz].

This POST request for updating the access rights of an Access Token SHOULD NOT include the parameters 'salt_input', 'context_id', 'client_cred' and 'client_cred_verify'. An exception is the case defined in Appendix A.3.6, where the Client, following a change of public key in the OSCORE group, requests a new Access Token associated to the new public key, while still without changing the existing pairwise OSCORE Security Context with the RS.

An example of such a request, with payload in CBOR diagnostic notation without the tag and value abbreviations is reported in Figure 10.

```
Header: POST (Code=0.02)
Uri-Host: "as.example.com"
Uri-Path: "token"
Content-Format: "application/ace+cbor"
Payload:
{
  "audience" : "tempSensor4711",
  "scope" : "read",
  "req_cnf" : {
    "kid" : h'01'
  }
}
```

Figure 10: Example C-to-AS POST /token request for updating rights to an Access Token bound to a symmetric key.

A.2.1.1. 'client_cred' Parameter

The 'client_cred' parameter is an OPTIONAL parameter of the Access Token request message defined in Section 5.8.1. of [I-D.ietf-ace-oauth-authz]. This parameter provides an asymmetric key that the Client wishes to use as its own public key, but which is not used as proof-of-possession key.

This parameter follows the syntax of the 'cnf' claim from Section 3.1 of [RFC8747] when including Value Type "COSE_Key" (1) and specifying an asymmetric key. Alternative Value Types defined in future specifications are fine to consider if indicating a non-encrypted asymmetric key.

A.2.2. AS-to-C: Access Token

After having verified the POST request to the /token endpoint and that the Client is authorized to obtain an Access Token corresponding to its Access Token request, the AS MUST verify the signature in the 'client_cred_verify' parameter, by using the public key specified in the 'client_cred' parameter. If the verification fails, the AS considers the Client request invalid. The AS does not perform this operation when asked to update a previously released Access Token.

If all verifications are successful, the AS responds as defined in Section 5.8.2 of [I-D.ietf-ace-oauth-authz]. If the Client request was invalid, or not authorized, the AS returns an error response as described in Section 5.8.3 of [I-D.ietf-ace-oauth-authz].

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

In particular, the Client MUST follow Appendix A.3.3 to derive the pairwise OSCORE Security Context to use for communications with the RS. Instead, the Client has already established the related Group OSCORE Security Context to communicate with members of the OSCORE group, upon previously joining that group.

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

In contrast with the main mode of this profile, the Access Token response to the Client is analogous to the one in the OSCORE profile of ACE, as described in Section 3.2 of [I-D.ietf-ace-oscore-profile]. In particular, the AS provides an OSCORE_Input_Material object, which is defined in Section 3.2.1 of [I-D.ietf-ace-oscore-profile] and included in the 'cnf' parameter (see Section 3.2 of [I-D.ietf-ace-oauth-params]) of the Access Token response.

The AS MUST send different OSCORE_Input_Material (and therefore different Access Tokens) to different authorized clients, in order for the RS to differentiate between clients.

In the issued Access Token, the AS MUST include as metadata the same information as defined in the main mode of this profile (see Section 3.2) with the following modifications.

- The public key that the client uses in the OSCORE group and specified in the 'client_cred' parameter of the Token request (see Appendix A.2.1) MUST also be included in the Access Token. If the Access Token is a CWT, the AS MUST include it in the 'client_cred' claim of the Access Token, defined in Appendix A.2.2.2 of this specification.

- The OSCORE_Input_Material specified in the 'cnf' parameter of the Access Token response MUST also be included in the Access Token. If the Access Token is a CWT, the same OSCORE_Input_Material included in the 'cnf' parameter of the Access Token response MUST be included in the 'osc' field (see Section 9.5 of [I-D.ietf-ace-oscore-profile]) of the 'cnf' claim of the Access Token.

Figure 11 shows an example of such an AS response, with payload in CBOR diagnostic notation without the tag and value abbreviations.

```
Header: Created (Code=2.01)
Content-Type: "application/ace+cbor"
Payload:
{
  "access_token" : h'8343a1010aa2044c53 ...'
   (remainder of CWT omitted for brevity),
  "profile" : "coap_group_oscore",
  "expires_in" : 3600,
  "cnf" : {
    "osc" : {
      "alg" : AES-CCM-16-64-128,
      "id"  : h'01',
      "ms" : h'f9af838368e353e78888e1426bd94e6f',
      "salt" : h'1122',
      "contextId" : h'99'
    }
  }
}
```

Figure 11: Example AS-to-C Access Token response with the Group OSCORE profile.

Figure 12 shows an example CWT, containing the necessary OSCORE parameters in the 'cnf' claim, in CBOR diagnostic notation without tag and value abbreviations.

```
{
  "aud" : "tempSensorInLivingRoom",
  "iat" : 1360189224,
  "exp" : 1360289224,
  "scope" :  "temperature_g firmware_p",
  "cnf" : {
    "osc" : {
      "alg" : AES-CCM-16-64-128,
      "id"  : h'01',
      "ms" : h'f9af838368e353e78888e1426bd94e6f',
      "salt" : h'1122',
      "contextId" : h'99'
    },
    "salt_input" : h'00',
    "contextId_input" : h'abcd0000',
    "client_cred" : {
      "COSE_Key" : {
        "kty" : EC2,
        "crv" : P-256,
        "x" : h'd7cc072de2205bdc1537a543d53c60a6acb62eccd890c7fa
                27c9e354089bbe13',
        "y" : h'f95e1d4b851a2cc80fff87d8e23f22afb725d535e515d020
                731e79a3b4e47120'
      }
    }
}
```

Figure 12: Example CWT with OSCORE parameters (CBOR diagnostic notation).

The same CWT as in Figure 12 and encoded in CBOR is shown in Figure 13, using the value abbreviations defined in [I-D.ietf-ace-oauth-authz] and [RFC8747].

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

```
A8                                      # map(8)
   03                                   # unsigned(3)
   76                                   # text(22)
      74656D7053656E736F72496E4C6976696E67526F6F6D
   06                                   # unsigned(6)
   1A 5112D728                          # unsigned(1360189224)
   04                                   # unsigned(4)
   1A 51145DC8                          # unsigned(1360289224)
   09                                   # unsigned(9)
   78 18                                # text(24)
```

```
   74656D7065726174747572655F67206669726D776172655F70
08                                  # unsigned(8)
A1                                  # map(1)
   04                               # unsigned(4)
   A5                               # map(5)
      04                            # unsigned(4)
      0A                            # unsigned(10)
      00                            # unsigned(0)
      41                            # bytes(1)
         01                         # "\x01"
      02                            # unsigned(2)
      50                            # bytes(16)
         F9AF838368E353E78888E1426BD94E6F
      05                            # unsigned(5)
      42                            # bytes(2)
         1122                       # "\x11\""
      06                            # unsigned(6)
      41                            # bytes(1)
         99                         # "\x99"
18 3C                               # unsigned(60)
41                                  # bytes(1)
   00
18 3D                               # unsigned(61)
44                                  # bytes(4)
   ABCD0000
18 3E                               # unsigned(62)
A1                                  # map(1)
   01                               # unsigned(1)
   A4                               # map(4)
      01                            # unsigned(1)
      02                            # unsigned(2)
      20                            # negative(0)
      01                            # unsigned(1)
      21                            # negative(1)
      58 20                         # bytes(32)
         D7CC072DE2205BDC1537A543D53C60A6ACB62ECCD890C7FA27C9
         E354089BBE13
      22                            # negative(2)
      58 20                         # bytes(32)
         F95E1D4B851A2CC80FFF87D8E23F22AFB725D535E515D020731E
         79A3B4E47120
```

Figure 13: Example CWT with OSCORE parameters.

If the Client has requested an update to its access rights using the same pairwise OSCORE Security Context, which is valid and authorized, the AS MUST omit the 'cnf' parameter in the response to the client.

Instead, the updated Access Token conveyed in the AS-to-C response MUST include a 'cnf' claim specifying a 'kid' field, as defined in Section 3.1 of [RFC8747]. The response from the AS MUST carry the OSCORE Input Material identifier in the 'kid' field within the 'cnf' claim of the Access Token. That is, the 'kid' field is a CBOR byte string, with value the same value of the 'kid' field of the 'req_cnf' parameter from the C-to-AS request for updating rights to the Access Token (see Figure 10). This information needs to be included in the Access Token, in order for the RS to identify the previously generated pairwise OSCORE Security Context.

Figure 14 shows an example of such an AS response, with payload in CBOR diagnostic notation without the tag and value abbreviations. The Access Token has been truncated for readability.

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

Figure 14: Example AS-to-C Access Token response with the Group OSCORE profile, for update of access rights.

Figure 15 shows an example CWT, containing the necessary OSCORE parameters in the 'cnf' claim for update of access rights, in CBOR diagnostic notation without tag and value abbreviations.

```
{
  "aud" : "tempSensorInLivingRoom",
  "iat" : 1360189224,
  "exp" : 1360289224,
  "scope" :  "temperature_h",
  "cnf" : {
    "kid" : h'01'
  }
}
```

Figure 15: Example CWT with OSCORE parameters for update of access rights.

A.2.2.1. Public Key Hash as Client Credential

As a possible optimization to limit the size of the Access Token, the AS may specify as value of the 'client_cred' claim simply the hash of the Client's public key. The specifically used hash-function MUST be collision-resistant on byte-strings, and MUST be selected from the "Named Information Hash Algorithm" Registry defined in Section 9.4 of [RFC6920].

In particular, the AS provides the Client with an Access Token as defined in Appendix A.2.2, with the following differences.

The AS prepares INPUT_HASH as follows, with | denoting byte string concatenation.

- If the public key has COSE Key Type OKP, INPUT_HASH = i, where 'i' is the x-parameter of the COSE_Key specified in the 'client_cred' parameter of the Token request, encoded as a CBOR byte string.

- If the public key has COSE Key Type EC2, INPUT_HASH = (i_1 | i_2), where 'i_1' and 'i_2' are the x-parameter and y-parameter of the COSE_Key specified in the 'client_cred' parameter of the Token request, respectively, each encoded as a CBOR byte string.

- If the public key has COSE Key Type RSA, INPUT_HASH = (i_1 | i_2), where 'i_1' and 'i_2' are the n-parameter and e-parameter of the COSE_Key specified in the 'client_cred' parameter of the Token request, respectively, each encoded as a CBOR byte string.

Then, the AS hashes INPUT_HASH according to the procedure described in [RFC6920], with the output OUTPUT_HASH in binary format, as described in Section 6 of [RFC6920].

Finally, the AS includes a single entry within the 'client_cred' claim of the Access Token. This entry has label "kid" (3) defined in Section 3.1 of [RFC8747], and value a CBOR byte string wrapping OUTPUT_HASH.

Upon receiving the Access Token, the RS processes it according to Appendix A.3.2, with the following differences.

The RS considers the content of the 'client_cred' claim as the hash of the public key associated to the signing private key that the Client uses in the OSCORE group, which is identified by the 'context_id' parameter.

The RS MAY additionally request the Group Manager of the OSCORE group for the public key of that Client, as described in

[I-D.ietf-ace-key-groupcomm-oscore], specifying as Sender ID of that Client in the OSCORE group the value of the 'salt_input' claim included in the Access Token.

In such a case, the RS MUST check that the hash of the key retrieved from the Group Manager matches the hash retrieved from the 'client_cred' claim of the Access Token. The RS MUST calculate the hash using the same method as the AS described above, and using the same hash function. The hash function used can be determined from the information conveyed in the 'client_cred' claim, as the procedure described in [RFC6920] also encodes the used hash function as metadata of the hash value.

A.2.2.2. Client Credential Claim

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

This parameter follows the syntax of the 'cnf' claim from Section 3.1 of [RFC8747] when including Value Type "COSE_Key" (1) and specifying an asymmetric key. Alternative Value Types defined in future specifications are fine to consider if indicating a non-encrypted asymmetric key.

A.3. Client-RS Communication

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

That is, the Client generates a nonce N1 and posts it to the RS, together with: an identifier ID1 unique in the sets of its own Recipient IDs from its pairwise OSCORE Security Contexts; and the Access Token that includes the material provisioned by the AS.

Then, the RS generates a nonce N2, and an identifier ID2 unique in the sets of its own Recipient IDs from its pairwise OSCORE Security Contexts. After that, the RS derives a pairwise OSCORE Security Context as described in Section 3.2 of [RFC8613]. In particular, it uses the two nonces and the two identifiers established with the Client, as well as two shared secrets together with additional pieces of information specified in the Access Token.

Both the client and the RS generate the pairwise OSCORE Security Context using the pop-key as part of the OSCORE Master Secret. In

addition, the derivation of the pairwise OSCORE Security Context takes as input also information related to the OSCORE group, i.e. the Master Secret and Group Identifier of the group, as well as the Sender ID of the Client in the group. Hence, the derived pairwise OSCORE Security Context is also securely bound to the Group OSCORE Security Context of the OSCORE Group. Thus, the proof-of-possession required to bind the Access Token to the Client occurs after the first OSCORE message exchange.

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

Besides, just as in the main mode (see Section 4), the RS is able to verify whether the Client has indeed the claimed Sender ID and public key in the OSCORE group.

A.3.1. C-to-RS POST to authz-info Endpoint

The Client MUST generate a nonce N1, an OSCORE Recipient ID (ID1), and post them to the /authz-info endpoint of the RS together with the Access Token, as defined in Section 4.1 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

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

If the Client has already posted a valid Access Token, has already established a pairwise OSCORE Security Context with the RS, and wants to update its access rights, the Client can do so by posting the new Access Token (retrieved from the AS and specifying the updated set of access rights) to the /authz-info endpoint.

The Client MUST protect the request using either the pairwise OSCORE Security Context established during the first Access Token exchange, or the Group OSCORE Security Context associated to that pairwise OSCORE Security Context.

In either case, the Client MUST only send the Access Token in the payload, i.e. no nonce or identifier are sent. After proper verification (see Section 4.2 of [I-D.ietf-ace-oscore-profile]), the new Access Token will supersede the old one at the RS, by replacing the corresponding authorization information. At the same time, the RS will maintain the same pairwise OSCORE Security Context and Group

OSCORE Security Context, as now both associated to the new Access Token.

A.3.2. RS-to-C: 2.01 (Created)

The RS MUST verify the validity of the Access Token as defined in Section 4.2, with the following modifications.

- If the POST request to /authz-info is not protected, the RS checks that the 'cnf' claim is included in the Access Token and that it contains an OSCORE_Input_Material object. Also, the RS checks that the 'salt_input', 'client_cred' and 'contextId_input' claims are included in the Access Token.

- If the POST request to /authz-info is protected with the pairwise OSCORE Security Context shared with the Client or with the Group OSCORE Security Context of the OSCORE group, i.e. the Client is requesting an update of access rights, the RS checks that the 'cnf' claim is included in the Access Token and that it contains only the 'kid' field.

- If the 'salt_input', 'client_cred' and 'contextId_input' claims are included in the Access Token, the RS extracts the content of 'client_cred'. Then, the RS considers that content as the public key associated to the signing private key of the Client in the OSCORE group, whose GID is specified in the 'contextId_input' claim. The RS can compare this public key with the public key of the claimed Client retrieved from the Group Manager (see Section 4.2).

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

If the Access Token is valid and further checks on its content are successful, the RS proceeds as follows.

In case the POST request to /authz-info was not protected, the RS MUST generate a nonce N2, an OSCORE Recipient ID (ID2), and include them in the 2.01 (Created) response to the Client, as defined in Section 4.2 of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

Instead, in case the POST request to /authz-info was protected, the RS MUST ignore any nonce and identifiers in the request, if any was sent. Then, the RS MUST check that the 'kid' field of the 'cnf' claim in the new Access Token matches the identifier of the OSCORE Input Material of a pairwise OSCORE Security Context such that:

- The pairwise OSCORE Security Context was used to protect the request, if this was protected with OSCORE; or

- The pairwise OSCORE Security Context is bound to the Group OSCORE Security Context used to protect the request, if this was protected with Group OSCORE.

If either verification is successful, the new Access Token supersedes the old one at the RS. Besides, the RS associates the new Access Token to the same pairwise OSCORE Security Context identified above, as also bound to a Group OSCORE Security Context. The RS MUST respond with a 2.01 (Created) response with no payload, protected with the same Security Context used to protect the request. In particular, no new pairwise OSCORE Security Context is established between the Client and the RS. If any verification fails, the RS MUST respond with a 4.01 (Unauthorized) error response.

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

A.3.3. OSCORE Setup - Client Side

Once having received the 2.01 (Created) response from the RS, following an unprotected POST request to the /authz-info endpoint, the Client MUST extract the nonce N2 from the 'nonce2' parameter, and the Client identifier from the 'ace_server_recipientid' parameter in the CBOR map of the response payload. Note that this identifier is used by C as Sender ID in the pairwise OSCORE Security Context to be established with the RS, and is different as well as unrelated to the Sender ID of C in the OSCORE group.

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

- The Client MUST set the ID Context of the pairwise OSCORE Security Context as the concatenation of: i) GID, i.e. the Group Identifier of the OSCORE group, as specified by the Client in the 'context_id' parameter of the Client-to-AS request; ii) the nonce N1; iii) the nonce N2; and iv) CID, i.e. the value in the contextId parameter of the OSCORE_Input_Material provided in the 'cnf' parameter of the Access Token response from the AS. The concatenation occurs in this order: ID Context = GID | N1 | N2 | CID, where | denotes byte string concatenation.

- The Client MUST set the updated Master Salt of the pairwise OSCORE Security Context as the concatenation of SaltInput, MSalt, the

nonce N1, the nonce N2 and GMSalt, where: i) SaltInput is the Sender ID that the Client has in the OSCORE group, which is known to the Client as a member of the OSCORE group; ii) MSalt is the (optional) Master Salt in the pairwise OSCORE Security Context (received from the AS in the Token); and iii) GMSalt is the (optional) Master Salt in the Group OSCORE Security Context, which is known to the Client as a member of the OSCORE group. The concatenation occurs in this order: Master Salt = SaltInput | MSalt | N1 | N2 | GMSalt, where | denotes byte string concatenation. Optional values, if not specified, are not included in the concatenation. The five parameters SaltInput, MSalt, N1, N2 and GMSalt are to be concatenated as encoded CBOR byte strings. An example of Master Salt construction using CBOR encoding is given in Figure 16.

```
SaltInput, MSalt, N1, N2 and GMSalt, in CBOR diagnostic notation:
      SaltInput = h'00'
      MSalt = h'f9af838368e353e78888e1426bd94e6f'
      N1 = h'018a278f7faab55a'
      N2 = h'25a8991cd700ac01'
      GMSalt = h'99'

SaltInput, MSalt, N1, N2 and GMSalt, as CBOR encoded byte strings:
      SaltInput = 0x4100
      MSalt = 0x50f9af838368e353e78888e1426bd94e6f
      N1 = 0x48018a278f7faab55a
      N2 = 0x4825a8991cd700ac01
      GMSalt = 0x4199

Master Salt = 0x41 00
                50 f9af838368e353e78888e1426bd94e6f
                48 018a278f7faab55a
                48 25a8991cd700ac01
                41 99
```

Figure 16: Example of Master Salt construction using CBOR encoding.

- The Client MUST set the Master Secret of the pairwise OSCORE Security Context to the concatenation of MSec and GMSec, where: i) MSec is the value of the 'ms' parameter in the OSCORE_Input_Material of the 'cnf' parameter, received from the AS in Appendix A.2.2; while ii) GMSec is the Master Secret of the Group OSCORE Security Context, which is known to the Client as a member of the OSCORE group.

- The Client MUST set the Recipient ID as ace_client_recipientid, sent as described in Appendix A.3.1.

- The Client MUST set the Sender ID as ace_server_recipientid, received as described in Appendix A.3.1.

- The Client MUST set the AEAD Algorithm, ID Context, HKDF, and OSCORE Version as indicated in the corresponding parameters received from the AS in Appendix A.2.2, if present in the OSCORE_Input_Material of the 'cnf' parameter. In case these parameters are omitted, the default values SHALL be used as described in Section 3.2 and 5.4 of [RFC8613].

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

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

If any of the expected parameters is missing (e.g., any of the mandatory parameters from the AS or the RS), or if ace_client_recipientid equals ace_server_recipientid (and as a consequence the Sender and Recipient Keys derived would be equal, see Section 3.3 of [RFC8613]), then the client MUST stop the exchange, and MUST NOT derive the pairwise OSCORE Security Context. The Client MAY restart the exchange, to get the correct security input material.

The Client can use this pairwise OSCORE Security Context to send requests to the RS protected with OSCORE. Besides, the Client can use the Group OSCORE Security Context for protecting unicast requests to the RS, or multicast requests to the OSCORE group including also the RS. Mutual authentication as group members is only achieved after the client has successfully verified the Group OSCORE protected response from the RS. Similarly, mutual authentication as OSCORE peers is only achieved after the client has successfully verified the OSCORE protected response from the RS, using the pairwise OSCORE Security Context.

Note that the ID Context of the pairwise OSCORE Security Context can be assigned by the AS, communicated and set in both the RS and Client after the exchange specified in this profile is executed. Subsequently, the Client and RS can update their ID Context by running a mechanism such as the one defined in Appendix B.2 of [RFC8613] if they both support it and are configured to do so. In that case, the ID Context in the pairwise OSCORE Security Context will not match the "contextId" parameter of the corresponding OSCORE_Input_Material. Running the procedure in Appendix B.2 of [RFC8613] results in the keying material in the pairwise OSCORE

Security Contexts of the Client and RS being updated. The Client can achieve the same result by re-posting the Access Token to the unprotected /authz-info endpoint at the RS, as described in Section 4.1 of [I-D.ietf-ace-oscore-profile], although without updating the ID Context.

A.3.4. OSCORE Setup - Resource Server Side

After validation of the Access Token as defined in Appendix A.3.2 and after sending the 2.01 (Created) response to an unprotected POST request to the /authz-info endpoint, the RS performs the following actions, in order to set up and fully derive the pairwise OSCORE Security Context created to communicate with the Client.

- The RS MUST set the ID Context of the pairwise OSCORE Security Context as the concatenation of: i) GID, i.e. the Group Identifier of the OSCORE group, as specified in the 'contextId' parameter of the OSCORE_Input_Material, in the 'cnf' claim of the Access Token received from the Client (see Appendix A.3.1); ii) the nonce N1; iii) the nonce N2; and iv) CID which is the value in the contextId parameter of the OSCORE_Input_Material provided in the 'cnf' claim of the Access Token. The concatenation occurs in this order: ID Context = GID | N1 | N2 | CID, where | denotes byte string concatenation.

- The RS MUST set the new Master Salt of the pairwise OSCORE Security Context as the concatenation of SaltInput, MSalt, the nonce N1, the nonce N2 and GMSalt, where: i) SaltInput is the Sender ID that the Client has in the OSCORE group, as specified in the 'salt_input' claim included in the Access Token received from the Client (see Appendix A.3.1); ii) MSalt is the (optional) Master Salt in the pairwise OSCORE Security Context as specified in the 'salt' parameter in the OSCORE_Input_Material of the 'cnf' claim, included in the Access Token received from the Client; and iii) GMSalt is the (optional) Master Salt in the Group OSCORE Security Context, which is known to the RS as a member of the OSCORE group. The concatenation occurs in this order: Master Salt = SaltInput | MSalt | N1 | N2 | GMSalt, where | denotes byte string concatenation. Optional values, if not specified, are not included in the concatenation. The same considerations for building the Master Salt, considering the inputs as encoded CBOR byte strings as in Figure 16, hold also for the RS.

- The RS MUST set the Master Secret of the pairwise OSCORE Security Context to the concatenation of MSec and GMSec, where: i) MSec is the value of the 'ms' parameter in the OSCORE_Input_Material of the 'cnf' claim, included in the Access Token received from the Client (see Appendix A.3.1); while ii) GMSec is the Master Secret

of the Group OSCORE Security Context, which is known to the RS as a member of the OSCORE group.

- The RS MUST set the Recipient ID as ace_server_recipientid, sent as described in Appendix A.3.2.

- The RS MUST set the Sender ID as ace_client_recipientid, received as described in Appendix A.3.2.

- The RS MUST set the AEAD Algorithm, ID Context, HKDF, and OSCORE Version from the corresponding parameters received from the Client in the Access Token (see Appendix A.3.1), if present in the OSCORE_Input_Material of the 'cnf' claim. In case these parameters are omitted, the default values SHALL be used as described in Section 3.2 and 5.4 of [RFC8613].

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

Once having completed the derivation above, the RS MUST associate the authorization information from the Access Token with the just established pairwise OSCORE Security Context. Furthermore, as defined in Section 4.2, the RS MUST associate the authorization information from the Access Token with the Group OSCORE Security Context.

Then, the RS uses this pairwise OSCORE Security Context to verify requests from and send responses to the Client protected with OSCORE, when this Security Context is used. If OSCORE verification fails, error responses are used, as specified in Section 8 of [RFC8613].

Besides, the RS uses the Group OSCORE Security Context to verify (multicast) requests from and send responses to the Client protected with Group OSCORE. When processing an incoming request protected with Group OSCORE, the RS MUST consider as valid public key of the Client only the public key specified in the stored Access Token. As defined in Appendix A.3.6, a change of public key in the group requires the Client to upload to the RS a new Access Token, specifying the new public key in the 'client_cred' claim.

If Group OSCORE verification fails, error responses are used, as specified in Section 8 and Section 9 of [I-D.ietf-core-oscore-groupcomm]. Additionally, for every incoming request, if OSCORE or Group OSCORE verification succeeds, the verification of access rights is performed as described in Appendix A.3.5.

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

The same considerations, related to the value of the ID Context changing, as in Appendix A.3.3 hold also for the RS.

A.3.5. Access Rights Verification

The RS MUST follow the procedures defined in Section 4.4.

Additionally, if the RS receives an OSCORE-protected request from a Client, the RS processes it according to [RFC8613].

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

The response code MUST be 4.01 (Unauthorized) if the RS has no valid Access Token for the Client.

A.3.6. Change of Client's Public Key in the Group

During its membership in the OSCORE group, the client might change the public key it uses in the group. When this happens, the Client uploads the new public key to the Group Manager, as defined in Section 11 of [I-D.ietf-ace-key-groupcomm-oscore].

After that, the Client may still have an Access Token previously uploaded to the RS, which is not expired yet and still valid to the best of the Client's knowledge. Then, in order to continue communicating with the RS, the Client MUST perform the following actions.

1. The Client requests a new Access Token to the AS, as defined in Appendix A.2.1 for the update of access rights, i.e. with the 'req_cnf' parameter including only a 'kid' field. In particular, when sending the POST request to the AS, the Client indicates:

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

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

* The new public key it has in the OSCORE group, as value of the 'client_cred' parameter.

* The proof-of-possession signature corresponding to the new public key, as value of the 'client_cred_verify' parameter.

* The same current or instead new set of access rights, as value of the 'scope' parameter.

2. After receiving the response from the AS (see Appendix A.2.2), the Client performs the same exchanges with the RS as defined in Appendix A.3, with the following difference: the POST request to /authz-info for uploading the new Access Token MUST be protected with the pairwise OSCORE Security Context shared with the RS.

When receiving the new Access Token, the RS performs the same steps defined in Appendix A.3.2. In particular, no new pairwise OSCORE Security Context is established between the Client and the RS.

A.4. Secure Communication with the AS

The same considerations for secure communication with the AS as defined in Section 5 hold.

A.5. Discarding the Security Context

The Client and the RS MUST follow what is defined in Section 6 of [I-D.ietf-ace-oscore-profile] about discarding the pairwise OSCORE Security Context.

Additionally, they MUST follow what is defined in the main mode of the profile (see Section 6), with respect to the Group OSCORE Security Context.

The Client or RS can acquire a new Group OSCORE Security Context, by re-joining the OSCORE group, e.g. by using the approach defined in [I-D.ietf-ace-key-groupcomm-oscore]. In such a case, the Client SHOULD request a new Access Token and post it to the RS, in order to establish a new pairwise OSCORE Security Context and bind it to the Group OSCORE Security Context obtained upon re-joining the group.

A.6. CBOR Mappings

The new parameters defined in this document MUST be mapped to CBOR types as specified in Figure 6, with the following addition, using the given integer abbreviation for the map key.

Parameter name	CBOR Key	Value Type
client_cred	TBD6	map

Figure 17: CBOR mappings for new parameters.

The new claims defined in this document MUST be mapped to CBOR types as specified in Figure 7, with the following addition, using the given integer abbreviation for the map key.

Claim name	CBOR Key	Value Type
client_cred	TBD7	map

Figure 18: CBOR mappings for new claims.

A.7. Security Considerations

The dual mode of this profile inherits the security considerations from the main mode (see Section 8), as well as from the security considerations of the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile]. Also, the security considerations about OSCORE [RFC8613] hold for the dual mode of this profile, as to the specific use of OSCORE.

A.8. Privacy Considerations

The same privacy considerations as defined in the main mode of this profile apply (see Section 9).

In addition, as this profile mode also uses OSCORE, the privacy considerations from [RFC8613] apply as well, as to the specific use of OSCORE.

Furthermore, this profile mode inherits the privacy considerations from the OSCORE profile of ACE [I-D.ietf-ace-oscore-profile].

Appendix B. Profile Requirements

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

- (Optional) discovery process of how the Client finds the right AS for an RS it wants to send a request to: Not specified.
- Communication protocol the Client and the RS must use: CoAP.
- Security protocol(s) the Client and RS must use: Group OSCORE, i.e. exchange of secure messages by using a pre-established Group OSCORE Security Context. The optional dual mode defined in Appendix A additionally uses OSCORE, i.e. establishment of a pairwise OSCORE Security Context and exchange of secure messages.
- How the Client and the RS mutually authenticate: Explicitly, by possession of a common Group OSCORE Security Context, and by either: usage of digital counter signatures embedded in messages, if protected with the group mode of Group OSCORE; or protection of messages with the pairwise mode of Group OSCORE, by using pairwise symmetric keys, derived from the asymmetric keys of the two peers exchanging the message. Note that the mutual authentication is not completed before the Client has verified an OSCORE or a Group OSCORE response using the corresponding security context.
- Content-format of the protocol messages: "application/ace+cbor".
- Proof-of-Possession protocol(s) and how to select one; which key types (e.g. symmetric/asymmetric) supported: Group OSCORE algorithms; distributed and verified asymmetric keys. In the optional dual mode defined in Appendix A: OSCORE algorithms; pre-established symmetric keys.
- profile identifier: coap_group_oscore
- (Optional) how the RS talks to the AS for introspection: HTTP/CoAP (+ TLS/DTLS/OSCORE).
- How the client talks to the AS for requesting a token: HTTP/CoAP (+ TLS/DTLS/OSCORE).
- How/if the authz-info endpoint is protected: Not protected.
- (Optional) other methods of token transport than the authz-info endpoint: Not specified.

Acknowledgments

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

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

Authors' Addresses

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

Email: marco.tiloca@ri.se

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

Email: rikard.hoglund@ri.se

Ludwig Seitz
Combitech
Djaeknegatan 31
Malmoe SE-21135 Malmoe
Sweden

Email: ludwig.seitz@combitech.se

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

Email: francesca.palombini@ericsson.com

ACE Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2021

M. Tiloca
RISE AB
L. Seitz
Combitech
F. Palombini
Ericsson AB
S. Echeverria
G. Lewis
CMU SEI

February 22, 2021

Notification of Revoked Access Tokens in the Authentication and Authorization for Constrained Environments (ACE) Framework draft-tiloca-ace-revoked-token-notification-04

Abstract

This document specifies a method of the Authentication and Authorization for Constrained Environments (ACE) framework, which allows an Authorization Server to notify Clients and Resource Servers (i.e., registered devices) about revoked Access Tokens. The method relies on resource observation for the Constrained Application Protocol (CoAP), with Clients and Resource Servers observing a Token Revocation List on the Authorization Server. Resulting unsolicited notifications of revoked Access Tokens complement alternative approaches such as token introspection, while not requiring additional endpoints on Clients and Resource Servers.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 26, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Terminology 4
2. Protocol Overview 5
3. Token Hash 7
4. The TRL Resource 9
4.1. Update of the TRL Resource 9
5. The TRL Endpoint 9
5.1. Full Query of the TRL 10
5.2. Diff Query of the TRL 11
6. Upon Registration 13
7. Notification of Revoked Tokens 14
8. Interaction Examples 14
8.1. Full Query with Observation 15
8.2. Diff Query with Observation 16
8.3. Full Query with Observation and Additional Diff Query . . 18
9. Security Considerations 21
10. IANA Considerations 22
10.1. Media Type Registrations 22
10.2. CoAP Content-Formats Registry 23
10.3. Token Revocation List Registry 23
10.4. Expert Review Instructions 24
11. References 24
11.1. Normative References 24
11.2. Informative References 26
Appendix A. Usage of the Series Transfer Pattern 26
Appendix B. Usage of the "Cursor" Pattern 28
B.1. Full Query Request 28
B.2. Full Query Response 28
B.3. Diff Query Request 29
B.4. Diff Query Response 29
B.4.1. Empty Collection 29

B.4.2. Cursor Not Specified in the Diff Query Request . . . 29
B.4.3. Cursor Specified in the Diff Query Request 30
B.4.4. TRL Parameters 32
Acknowledgments 33
Authors' Addresses 33

1. Introduction

Authentication and Authorization for Constrained Environments (ACE) [I-D.ietf-ace-oauth-authz] is a framework that enforces access control on IoT devices acting as Resource Servers. In order to use ACE, both Clients and Resource Servers have to register with an Authorization Server and become a registered device. Once registered, a Client can send a request to the Authorization Server, to obtain an Access Token for a Resource Server. For a Client to access the Resource Server, the Client must present the issued Access Token at the Resource Server, which then validates and stores it.

Even though Access Tokens have expiration times, there are circumstances by which an Access Token may need to be revoked before its expiration time, such as: (1) a registered device has been compromised, or is suspected of being compromised; (2) a registered device is decommissioned; (3) there has been a change in the ACE profile for a registered device; (4) there has been a change in access policies for a registered device; and (5) there has been a change in the outcome of policy evaluation for a registered device (e.g., if policy assessment depends on dynamic conditions in the execution environment, the user context, or the resource utilization).

As discussed in Section 6.1 of [I-D.ietf-ace-oauth-authz], only client-initiated revocation is currently specified [RFC7009] for OAuth 2.0 [RFC6749], based on the assumption that Access Tokens in OAuth are issued with a relatively short lifetime. However, this may not be the case for constrained, intermittently connected devices, that need Access Tokens with relatively long lifetimes.

This document specifies a method for allowing registered devices to access and observe a Token Revocation List (TRL) resource on the Authorization Server, in order to get an updated list of revoked, but yet not expired, pertaining Access Tokens. In particular, registered devices rely on resource observation [RFC7641] for the Constrained Application Protocol (CoAP) [RFC7252]. The benefits of this method are that it complements token introspection and does not require any additional endpoints on the registered devices.

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 [I-D.ietf-ace-oauth-authz], as well as with terms and concepts related to CBOR Web Tokens (CWTs) [RFC8392], and JSON Web Tokens (JWTs) [RFC7519]. The terminology for entities in the considered architecture is defined in OAuth 2.0 [RFC6749]. In particular, this includes Client, Resource Server, and Authorization Server.

Readers are also expected to be familiar with the terms and concepts related to CBOR [RFC8949], JSON [RFC8259], the CoAP protocol [RFC7252], CoAP Observe [RFC7641], and the use of hash functions to name objects as defined in [RFC6920].

Note that, unless otherwise indicated, the term "endpoint" is used here following its OAuth definition, aimed at denoting resources such as /token and /introspect at the Authorization Server, and /authz-info at the Resource Server. This document does not use the CoAP definition of "endpoint", which is "An entity participating in the CoAP protocol."

This specification also refers to the following terminology.

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

- Token Revocation List (TRL): a collection of token hashes, in which the corresponding Access Tokens have been revoked but are not expired yet.

- TRL resource: a resource on the Authorization Server, with a TRL as its representation.

- TRL endpoint: an endpoint at the Authorization Server associated to the TRL resource. The default name of the TRL endpoint in a url-path is '/revoke/trl'. Implementations are not required to use this name, and can define their own instead.

- Registered device: a device registered at the Authorization Server, i.e. as a Client, or a Resource Server, or both. A registered device acts as caller of the TRL endpoint.

- Administrator: entity authorized to get full access to the TRL at the Authorization Server, and acting as caller of the TRL endpoint. An administrator is not necessarily a registered device as defined above, i.e. a Client requesting Access Tokens or a Resource Server consuming Access Tokens. How the administrator authorization is established and verified is out of the scope of this specification.

- Pertaining Access Token:

  * With reference to an administrator, an Access Token issued by the Authorization Server.

  * With reference to a registered device, an Access Token intended to be owned by that device. An Access Token pertains to a Client if the Authorization Server has issued the Access Token and provided it to that Client. An Access Token pertains to a Resource Server if the Authorization Server has issued the Access Token to be consumed by that Resource Server.

2. Protocol Overview

This protocol defines how a CoAP-based Authorization Server informs Clients and Resource Servers, i.e. registered devices, about revoked Access Tokens. How the relationship between the registered device and the Authorization Server is established is out of the scope of this specification.

At a high level, the steps of this protocol are as follows.

- Upon startup, the Authorization Server creates a TRL resource. At any point in time, the TRL resource represents the list of all revoked Access Tokens issued by the Authorization Server that are yet not expired.

- When a device registers at the Authorization Server, it receives the url-path to the TRL resource.

  After the registration procedure is finished, the registered device sends an Observation Request to that TRL resource as described in [RFC7641], i.e. a GET request with an Observe option set to 0 (register). Upon receiving the request, the Authorization Server adds the registered device to the list of observers of the TRL resource.

At any time, the registered device can send a GET request to the TRL endpoint. When doing so, it can request for: the current list of pertaining revoked Access Tokens (see Section 5.1); or the most recent TRL updates occurred over the list of pertaining revoked Access Tokens (see Section 5.2). In either case, the registered devices may especially rely on an Observation Request.

- When an Access Token is revoked, the Authorization Server adds the corresponding token hash to the TRL. Also, when a revoked Access Token eventually expires, the Authorization Server removes the corresponding token hash from the TRL.

  In either case, after updating the TRL, the Authorization Server sends Observe Notifications as per [RFC7641]. That is, one Observe Notification is sent to each registered device the Access Token pertains to, and specifies the current updated list of token hashes in the portion of the TRL pertaining to that device.

  Further Observe Notifications may be sent, consistently with ongoing additional observations of the TRL resource.

- An administrator can observe and access the TRL like a registered device, while getting the full updated representation of the TRL.

Figure 1 shows a high-level overview of the service provided by this protocol. In particular, it shows the Observe Notifications sent by the Authorization Server to one administrator and four registered devices, upon revocation of the issued Access Tokens t1, t2 and t3, with token hash th1, th2 and th3, respectively. Each dotted line associated to a pair of registered devices indicates the Access Token that they both own.

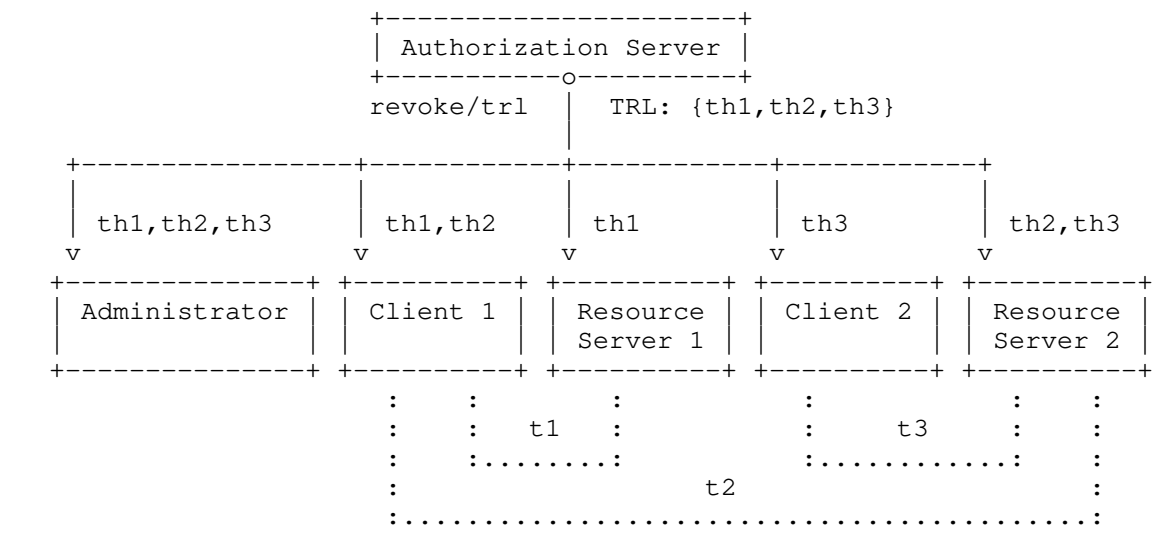


Figure 1: Protocol Overview

Section 8 provides examples of the protocol flow and message exchange between the Authorization Server and a registered device.

3. Token Hash

The token hash of an Access Token is computed as follows.

1. The Authorization Server defines ENCODED_TOKEN, as the content of the 'access_token' parameter in the Authorization Server response (see Section 5.8.2 of [I-D.ietf-ace-oauth-authz]), where the Access Token was included and returned to the requesting Client.

   Note that the content of the 'access_token' parameter is either:

   * A CBOR byte string, if the Access Token was transported using CBOR. With reference to the example in Figure 2, and assuming the string's length in bytes to be 119 (i.e., 0x77 in hexadecimal), then ENCODED_TOKEN takes the bytes {0x58 0x77 0xd0 0x83 0x44 0xa1 ...}, i.e. the raw content of the parameter 'access_token'.

   * A text string, if the Access Token was transported using JSON. With reference to the example in Figure 3, ENCODED_TOKEN takes "2YotnFZFEjr1zCsicMWpAA", i.e. the raw content of the parameter 'access_token'.

2. The Authorization Server defines HASH_INPUT as follows.

   * If CBOR was used to transport the Access Token (as a CWT or JWT), HASH_INPUT takes the same value of ENCODED_TOKEN.

   * If JSON was used to transport the Access Token (as a CWT or JWT), HASH_INPUT takes the serialization of ENCODED_TOKEN.

     In either case, HASH_INPUT results in the binary representation of the content of the 'access_token' parameter from the Authorization Server response.

3. The Authorization Server generates a hash value of HASH_INPUT 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 specifically used hash function MUST be collision-resistant on byte-strings, and MUST be selected from the "Named Information Hash Algorithm" Registry [Named.Information.Hash.Algorithm].

   The Authorization Server specifies the used hash function to registered devices during their registration procedure (see Section 6).

```
2.01 Created
Content-Format: application/ace+cbor
Max-Age: 85800
Payload:
{
   access_token : h'd08344a1...'
   (remainder of the Access Token omitted for brevity)
   token_type : pop,
   expires_in : 86400,
   profile    : coap_dtls,
   (remainder of the response omitted for brevity)
}
```

Figure 2: Example of Authorization Server response using CBOR

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
Payload:
{
   "access_token" : "2YotnFZFEjr1zCsicMWpAA"
   (remainder of the Access Token omitted for brevity)
   "token_type" : "pop",
   "expires_in" : 86400,
   "profile"    : "coap_dtls",
   (remainder of the response omitted for brevity)
}
```

Figure 3: Example of Authorization Server response using JSON

4. The TRL Resource

Upon startup, the Authorization Server creates a single TRL resource, encoded as a CBOR array.

Each element of the array is a CBOR byte string, with value the token hash of an Access Token. The order of the token hashes in the CBOR array is irrelevant, and the CBOR array MUST be treated as a set in which the order has no significant meaning.

The TRL is initialized as empty, i.e. the initial content of the TRL resource representation MUST be an empty CBOR array.

4.1. Update of the TRL Resource

The Authorization Server updates the TRL in the following two cases.

- When a non-expired Access Token is revoked, the token hash of the Access Token is added to the TRL resource representation. That is, a CBOR byte string with the token hash as its value is added to the CBOR array used as TRL resource representation.

- When a revoked Access Token expires, the token hash of the Access Token is removed from the TRL resource representation. That is, the CBOR byte string with the token hash as its value is removed from the CBOR array used as TRL resource representation.

5. The TRL Endpoint

Consistent with Section 6.5 of [I-D.ietf-ace-oauth-authz], all communications between a caller of the TRL endpoint and the Authorization Server MUST be encrypted, as well as integrity and

replay protected. Furthermore, responses from the Authorization Server to the caller MUST be bound to the caller's request.

The Authorization Server MUST implement measures to prevent access to the TRL endpoint by entities other than registered devices and authorized administrators.

The TRL endpoint supports only the GET method, and allows two types of query of the TRL.

- Full query: the Authorization Server returns the token hashes of the revoked Access Tokens currently in the TRL and pertaining to the requester. The Authorization Server MUST support this type of query. The processing of a full query and the related response format are defined in Section 5.1.

- Diff query: the Authorization Server returns a list of diff entries. Each diff entry is related to one of the most recent updates, in the portion of the TRL pertaining to the requester. The Authorization Server MAY support this type of query.

  The entry associated to one of such updates contains a list of token hashes, such that: i) the corresponding revoked Access Tokens pertain to the requester; and ii) they were added to or removed from the TRL at that update. The processing of a diff query and the related response format are defined in Section 5.2.

The TRL endpoint allows the following query parameter in a GET request.

- 'diff': if included, it indicates to perform a diff query of the TRL. Its value MUST be either:

  * the integer 0, indicating that a (notification) response should include as many diff entries as the Authorization Server can provide in the response; or

  * a positive integer greater than 0, indicating the maximum number of diff entries that a (notification) response should include.

5.1. Full Query of the TRL

In order to produce a (notification) response to a GET request asking for a full query of the TRL, the Authorization Server performs the following actions.

1. From the current TRL resource representation, the Authorization Server builds a set HASHES, such that:

   * If the requester is a registered device, HASHES specifies the token hashes of the Access Tokens pertaining to that registered device. The Authorization Server can use the authenticated identity of the registered device to perform the necessary filtering on the TRL resource representation.

   * If the requester is an administrator, HASHES specifies all the token hashes in the current TRL resource representation.

2. The Authorization Server sends a 2.05 (Content) Response to the requester, with a CBOR array as payload. Each element of the array specifies one of the token hashes from the set HASHES, encoded as a CBOR byte string.

   The order of the token hashes in the CBOR array is irrelevant, i.e. the CBOR array MUST be treated as a set in which the order has no significant meaning.

5.2. Diff Query of the TRL

In order to produce a (notification) response to a GET request asking for a diff query of the TRL, the Authorization Server performs the following actions.

1. The Authorization Server defines the positive integer NUM. If the value N specified in the query parameter 'diff' of the GET request is equal to 0 or greater than a pre-defined positive integer N_MAX, then NUM takes the value of N_MAX. Otherwise, NUM takes N.

2. The Authorization Server prepares U = min(NUM, SIZE) diff entries, where SIZE <= N_MAX is the number of TRL updates pertaining to the requester and currently stored at the Authorization Server. That is, the diff entries are related to the U most recent TRL updates pertaining to the requester. In particular, the first entry refers to the most recent of such updates, the second entry refers to the second from last of such updates, and so on.

   Each diff entry is a CBOR array 'diff-entry', which includes the following two elements.

   * The first element is a CBOR array 'removed'. Each element of the array is a CBOR byte string, with value the token hash of an Access Token such that: it pertained to the requester; and

it was removed from the TRL during the update associated to the diff entry.

* The second element is a CBOR array 'added'. Each element of the array is a CBOR byte string, with value the token hash of an Access Token such that: it pertains to the requester; and it was added to the TRL during the update associated to the diff entry.

The order of the token hashes in the CBOR arrays 'removed' and 'added' is irrelevant. That is, the CBOR arrays 'removed' and 'added' MUST be treated as a set in which the order of elements has no significant meaning.

3. The Authorization Server prepares a 2.05 (Content) Response for the requester, with a CBOR array 'diff' of U elements as payload. Each element of the CBOR array 'diff' specifies one of the CBOR arrays 'diff-entry' prepared at point 2 as diff entries.

   Within the CBOR array 'diff', the CBOR arrays 'diff-entry' MUST be sorted to reflect the corresponding updates to the TRL in reverse chronological order. That is, the first 'diff-entry' element of 'diff' relates to the most recent update to the portion of the TRL pertaining to the requester.

The CDDL definition [RFC8610] of the CBOR array 'diff' formatted as in the response from the Authorization Server is provided below.

```
token-hash = bytes
trl-patch = [* token-hash]
diff-entry = [removed: trl-patch, added: trl-patch]
diff = [* diff-entry]
```

Figure 4: CDDL definition of the response payload following a Diff Query request to the TRL endpoint

If the Authorization Server supports diff queries:

- The Authorization Server MUST return a 4.00 (Bad Request) response in case the 'diff' parameter specifies a value other than 0 or than a positive integer.

- The Authorization Server MUST keep track of N_MAX most recent updates to the portion of the TRL that pertains to each caller of the TRL endpoint. The particular method to achieve this is implementation-specific.

- When SIZE is equal to N_MAX, and a new TRL update occurs as pertaining to a registered device, the Authorization Server MUST first delete the oldest stored update for that device, before storing this latest update as the most recent one for that device.

- The Authorization Server SHOULD provide registered devices and administrators with the value of N_MAX, upon their registration (see Section 6).

If the Authorization Server does not support diff queries, it proceeds as when processing a full query (see Section 5.1).

Appendix A discusses how the diff query of the TRL is in fact a usage example of the Series Transfer Pattern defined in [I-D.bormann-t2trg-stp].

Appendix B discusses how the diff query of the TRL can be further improved by using the "Cursor" pattern defined in Section 3.3 of [I-D.bormann-t2trg-stp].

6. Upon Registration

During the registration process at the Authorization Server, an administrator or a registered device receives the following information as part of the registration response.

- The url-path to the TRL endpoint at the Authorization Server.

- The hash function used to compute token hashes. This is specified as an integer or a text string, taking value from the "ID" or "Hash Name String" column of the "Named Information Hash Algorithm" Registry [Named.Information.Hash.Algorithm], respectively.

- Optionally, a positive integer N_MAX, if the Authorization Server supports diff queries of the TRL resource (see Section 5.2).

After the registration procedure is finished, the administrator or registered device performs a GET request to the TRL resource, including the CoAP Observe option set to 0 (register), in order to start an observation of the TRL resource at the Authorization Server, as per Section 3.1 of [RFC7641]. The GET request can express the wish for a full query (see Section 5.1) or a diff query (see Section 5.2) of the TRL.

In case the request is successfully processed, The Authorization Server replies using the CoAP response code 2.05 (Content) and including the CoAP Observe option in the response. The payload of

the response is formatted as defined in Section 5.1 or in Section 5.2, in case the GET request was for a full query or a diff query of the TRL, respectively.

Further details about the registration process at the Authorization Server are out of scope for this specification. Note that the registration process is also out of the scope of the ACE framework for Authentication and Authorization (see Section 5.5 of [I-D.ietf-ace-oauth-authz]).

7. Notification of Revoked Tokens

When the TRL is updated (see Section 4.1), the Authorization Server sends Observe Notifications to every observer of the TRL resource. Observe Notifications are sent as per Section 4.2 of [RFC7641].

The payload of each Observe Notification is formatted as defined in Section 5.1 or in Section 5.2, in case the original Observation Request was for a full query or a diff query of the TRL, respectively.

Furthermore, an administrator or a registered device can send additional GET requests to the TRL endpoint at any time, in order to retrieve the token hashes of the pertaining revoked Access Tokens. When doing so, the caller of the TRL endpoint can perform a full query (see Section 5.1) or a diff query (see Section 5.2).

8. Interaction Examples

This section provides examples of interactions between a Resource Server RS as registered device and an Authorization Server AS. The Authorization Server supports both full query and diff query of the TRL, as defined in Section 5.1 and Section 5.2, respectively.

The details of the registration process are omitted, but it is assumed that the Resource Server sends an unspecified payload to the Authorization Server, which replies with a 2.01 (Created) response.

The payload of the registration response is a CBOR map, which includes the following entries:

- a "trl" parameter, specifying the path of the TRL resource;
- a "trl_hash" parameter, specifying the hash function used to computed token hashes as defined in Section 3;
- an "n_max" parameter, specifying the value of N_MAX, i.e. the maximum number of TRL updates pertaining to each registered device

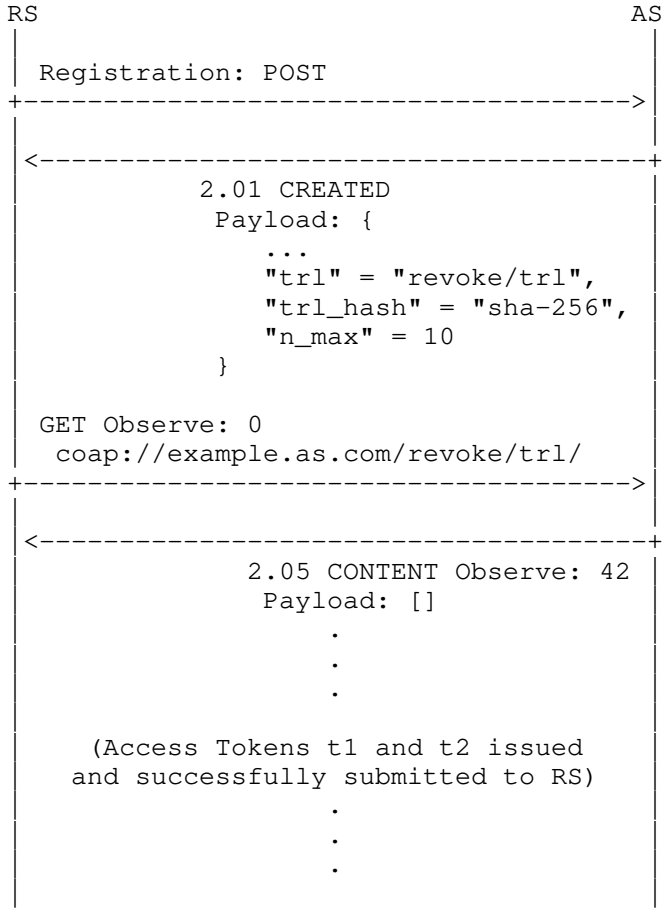
that the Authorization Server retains for that device (see Section 5.2);

- possible further parameters related to the registration process.

Furthermore, 'h(x)' refers to the hash function used to compute the token hashes, as defined in Section 3 of this specification and according to [RFC6920]. Assuming the usage of CWTs transported in CBOR, 'bstr.h(t1)' and 'bstr.h(t2)' denote the byte-string representations of the token hashes for the Access Tokens t1 and t2, respectively.

8.1. Full Query with Observation

Figure 5 shows an example interaction considering a CoAP observation and a full query of the TRL.



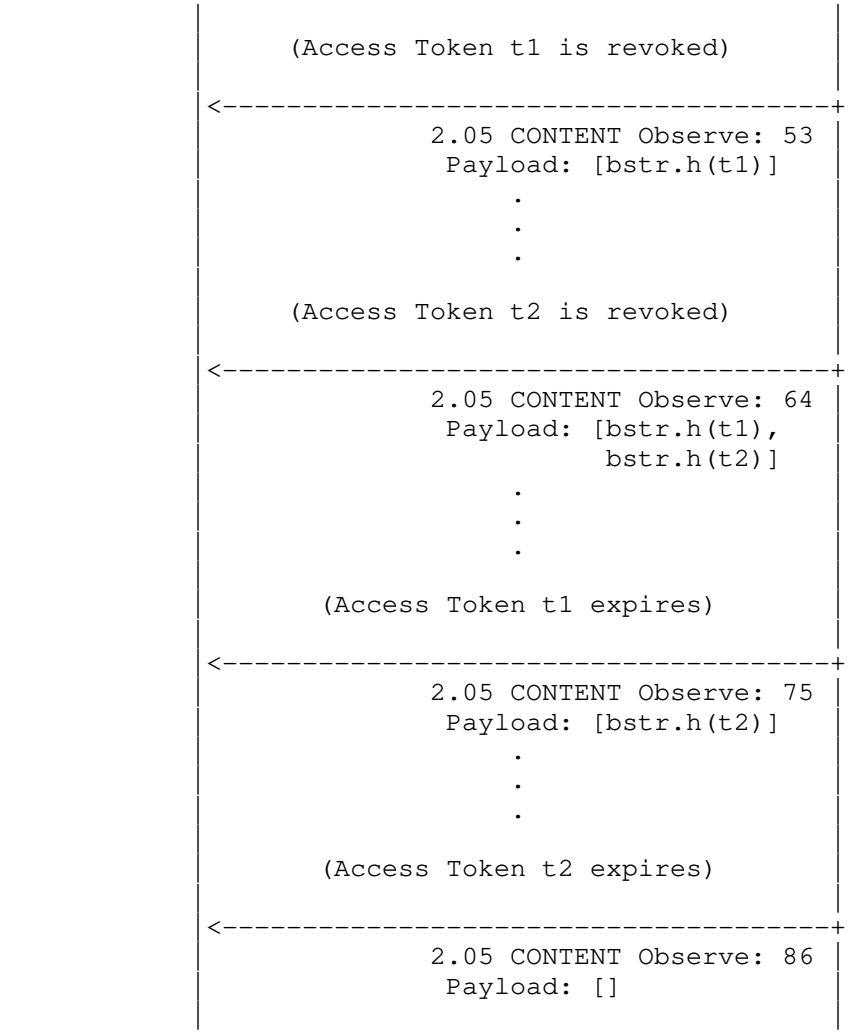
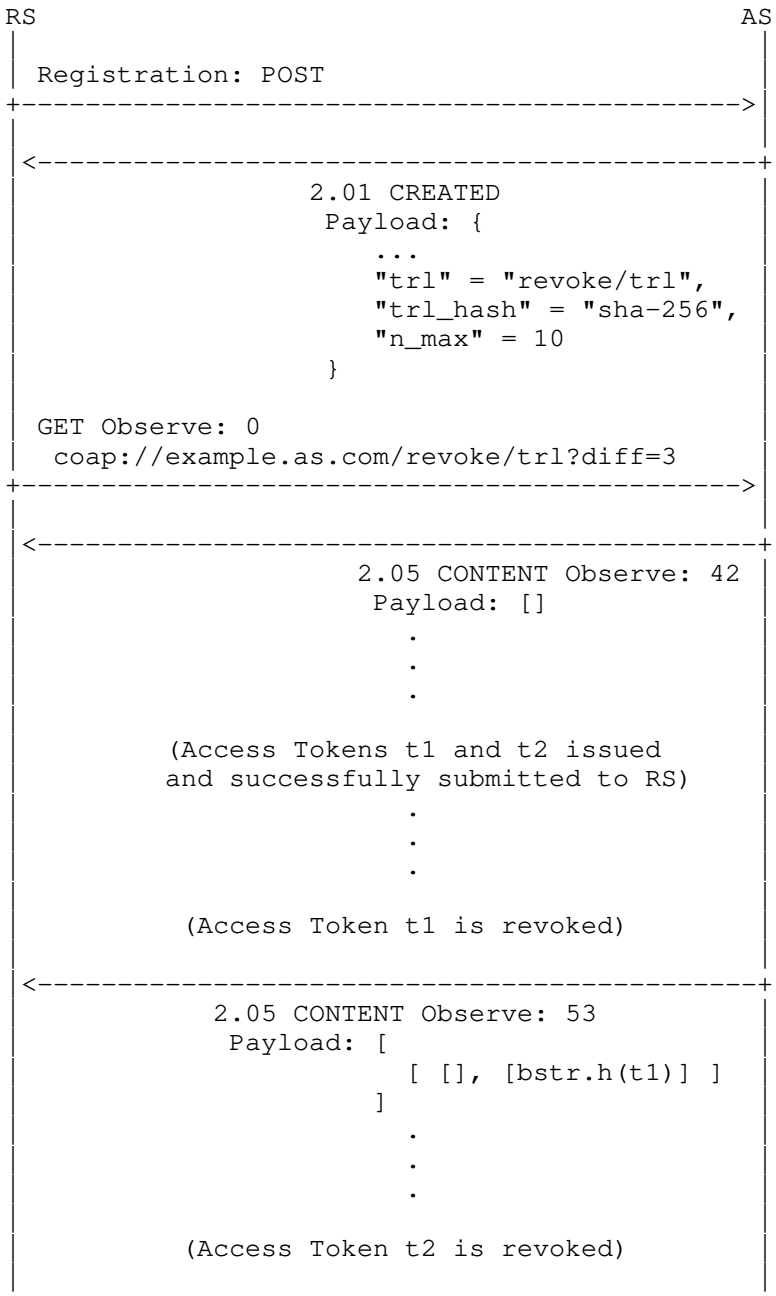


Figure 5: Interaction for Full Query with Observation

8.2. Diff Query with Observation

Figure 6 shows an example interaction considering a CoAP observation and a diff query of the TRL.

The Resource Server indicates N=3 as value of the query parameter "diff", i.e. as the maximum number of diff entries to be specified in a response from the Authorization Server.


RS
AS
Registration: POST
2.01 CREATED
Payload: {
...
"trl" = "revoke/trl",
"trl_hash" = "sha-256",
"n_max" = 10
}
GET Observe: 0
coap://example.as.com/revoke/trl?diff=3
2.05 CONTENT Observe: 42
Payload: []
(Access Tokens t1 and t2 issued
and successfully submitted to RS)
(Access Token t1 is revoked)
2.05 CONTENT Observe: 53
Payload: [
[[], [bstr.h(t1)]]
]
(Access Token t2 is revoked)

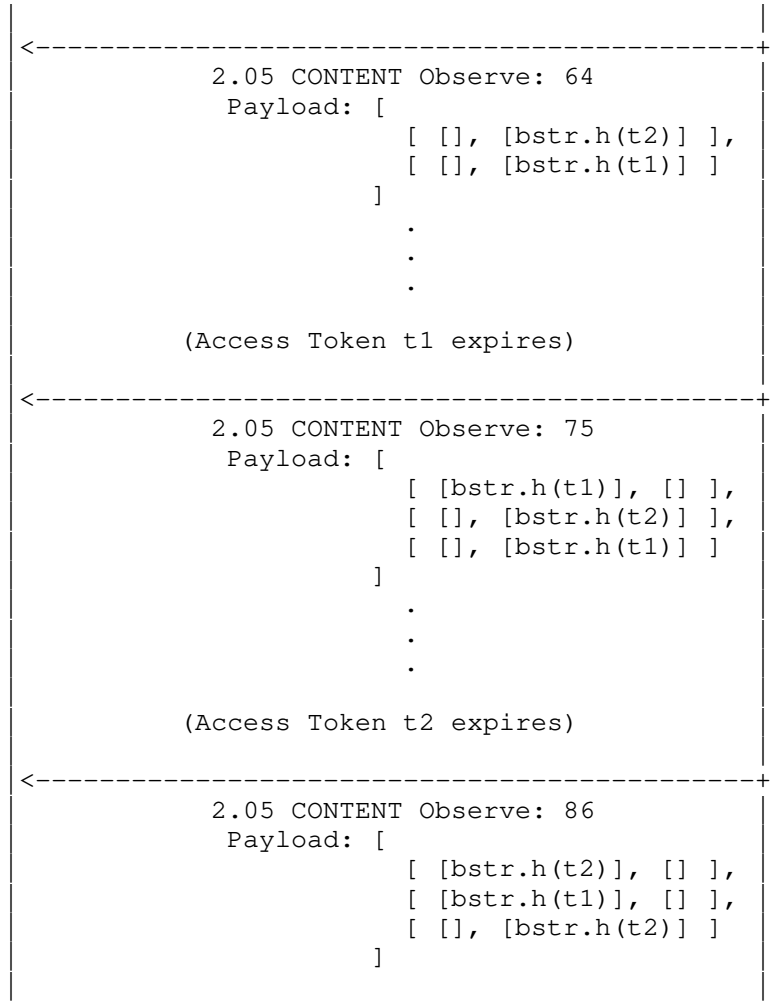


Figure 6: Interaction for Diff Query with Observation

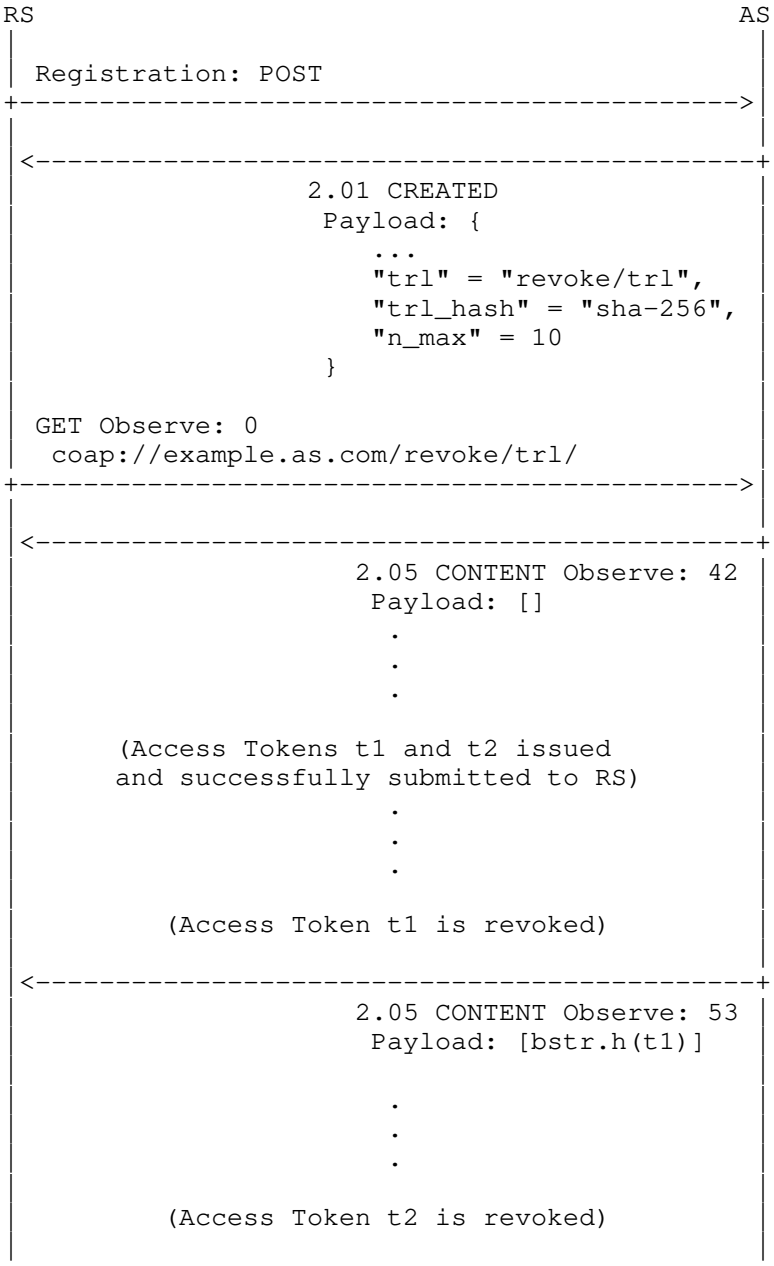
8.3. Full Query with Observation and Additional Diff Query

Figure 7 shows an example interaction considering a CoAP observation and a full query of the TRL.

The example also considers one of the notifications from the Authorization Server to get lost in transmission, and thus not reaching the Resource Server.

When this happens, and after a waiting time defined by the application has elapsed, the Resource Server sends a GET request with

no observation to the Authorization Server, to perform a diff query of the TRL. The Resource Server indicates N=8 as value of the query parameter "diff", i.e. as the maximum number of diff entries to be specified in a response from the Authorization Server.



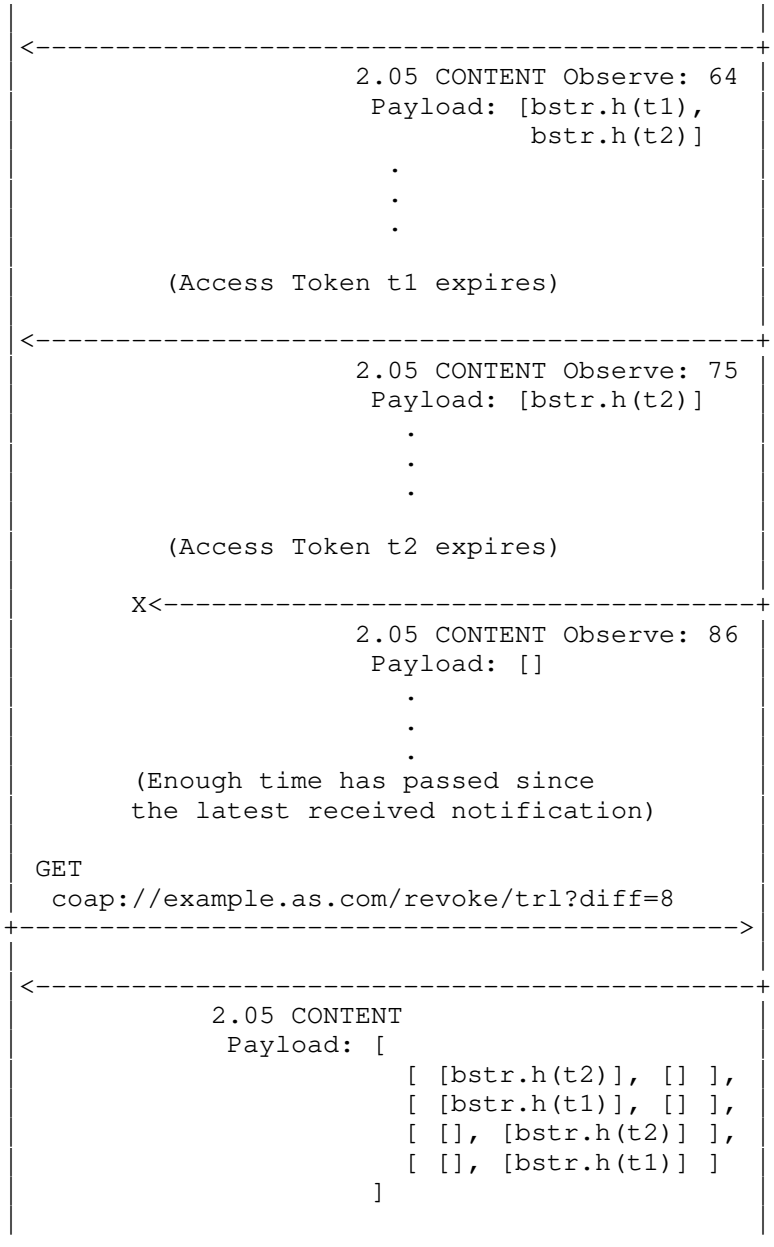


Figure 7: Interaction for Full Query with Observation and Diff Query

9. Security Considerations

Security considerations are inherited from the ACE framework for Authentication and Authorization [I-D.ietf-ace-oauth-authz], from [RFC8392] as to the usage of CWTs, from [RFC7519] as to the usage of JWTs, from [RFC7641] as to the usage of CoAP Observe, and from [RFC6920] with regards to resource naming through hashes. The following considerations also apply.

The Authorization Server MUST ensure that each registered device can access and retrieve only its pertaining portion of the TRL. To this end, the Authorization Server can perform the required filtering based on the authenticated identity of the registered device, i.e., a (non-public) identifier that the Authorization Server can securely relate to the registered device and the secure association they use to communicate.

Disclosing any information about revoked Access Tokens to entities other than the intended registered devices may result in privacy concerns. Therefore, the Authorization Server MUST ensure that, other than registered devices accessing their own pertaining portion of the TRL, only authorized and authenticated administrators can retrieve the full TRL. To this end, the Authorization Server may rely on an access control list or similar.

If a registered device has many non-expired Access Tokens associated to itself that are revoked, the pertaining portion of the TRL could grow to a size bigger than what the registered device is prepared to handle upon reception, especially if relying on a full query of the TRL resource (see Section 5.1). This could be exploited by attackers to negatively affect the behavior of a registered device. Short expiration times could help reduce the size of a TRL, but an Authorization Server SHOULD take measures to limit this size.

Most of the communication about revoked Access Tokens presented in this specification relies on CoAP Observe Notifications sent from the Authorization Server to a registered device. The suppression of those notifications by an external attacker that has access to the network would prevent registered devices from ever knowing that their pertaining Access Tokens have been revoked. To avoid this, a registered device SHOULD NOT rely solely on the CoAP Observe notifications. In particular, a registered device SHOULD also regularly poll the Authorization Server for the most current information about revoked Access Tokens, by sending GET requests to the TRL endpoint according to an application policy.

10. IANA Considerations

This document has the following actions for IANA.

10.1. Media Type Registrations

This specification registers the 'application/ace-trl+cbor' media type for messages of the protocols defined in this document encoded in CBOR. This registration follows the procedures specified in [RFC6838].

Type name: application

Subtype name: ace-trl+cbor

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Must be encoded as CBOR map containing the protocol parameters defined in [this document].

Security considerations: See Section 9 of this document.

Interoperability considerations: N/A

Published specification: [this document]

Applications that use this media type: The type is used by Authorization Servers, Clients and Resource Servers that support the notification of revoked Access Tokens, according to a Token Revocation List maintained by the Authorization Server as specified in [this document].

Fragment identifier considerations: N/A

Additional information: N/A

Person & email address to contact for further information: <iesg@ietf.org>

Intended usage: COMMON

Restrictions on usage: None

Author: Marco Tiloca <marco.tiloca@ri.se.com>

Change controller: IESG

10.2. CoAP Content-Formats Registry

This specification registers the following entry to the "CoAP Content-Formats" registry, within the "CoRE Parameters" registry:

Media Type: application/ace-trl+cbor

Encoding: -

ID: TBD

Reference: [this document]

10.3. Token Revocation List Registry

This specification establishes the "Token Revocation List" IANA Registry. The Registry has been created to use the "Expert Review" registration procedure [RFC8126]. Expert review guidelines are provided in Section 10.4. It should be noted that, in addition to the expert review, some portions of the Registry require a specification, potentially a Standards Track RFC, to be supplied as well.

The columns of this Registry are:

- Name: This is a descriptive name that enables easier reference to the item. The name MUST be unique. It is not used in the encoding.

- CBOR Key: This is the value used as CBOR key of the item. These values MUST be unique. The value can be a positive integer or a negative integer. Different ranges of values use different registration policies [RFC8126]. Integer values from -256 to 255 are designated as Standards Action. Integer values from -65536 to -257 and from 256 to 65535 are designated as Specification Required. Integer values greater than 65535 are designated as Expert Review. Integer values less than -65536 are marked as Private Use.

- CBOR Type: This contains the CBOR type of the item, or a pointer to the registry that defines its type, when that depends on another item.

- Reference: This contains a pointer to the public specification for the item.

This Registry has been initially populated by the values in Appendix B.4.4. The Reference column for all of these entries refers to this document.

10.4. Expert Review Instructions

The IANA registry established in this document is defined as expert review. This section gives some general guidelines for what the experts should be looking for, but they are being designated as experts for a reason so they should be given substantial latitude.

Expert reviewers should take into consideration the following points:

- Point squatting should be discouraged. Reviewers are encouraged to get sufficient information for registration requests to ensure that the usage is not going to duplicate one that is already registered and that the point is likely to be used in deployments. The zones tagged as private use are intended for testing purposes and closed environments, code points in other ranges should not be assigned for testing.

- Specifications are required for the standards track range of point assignment. Specifications should exist for specification required ranges, but early assignment before a specification is available is considered to be permissible. Specifications are needed for the first-come, first-serve range if they are expected to be used outside of closed environments in an interoperable way. When specifications are not provided, the description provided needs to have sufficient information to identify what the point is being used for.

- Experts should take into account the expected usage of fields when approving point assignment. The fact that there is a range for standards track documents does not mean that a standards track document cannot have points assigned outside of that range. The length of the encoded value should be weighed against how many code points of that length are left, the size of device it will be used on, and the number of code points left that encode to that size.

11. References

11.1. Normative References

[I-D.ietf-ace-oauth-authz]
Seitz, L., Selander, G., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)", draft-ietf-ace-oauth-authz-37 (work in progress), February 2021.

[Named.Information.Hash.Algorithm]
IANA, "Named Information Hash Algorithm", <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/info/rfc2119>.

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

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[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/info/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/info/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/info/rfc7519>.

[RFC7641] Hartke, K., "Observing Resources in the Constrained Application Protocol (CoAP)", RFC 7641, DOI 10.17487/RFC7641, September 2015, <https://www.rfc-editor.org/info/rfc7641>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

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

[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/info/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/info/rfc8392>.

[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/info/rfc8610>.

[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/info/rfc8949>.

11.2. Informative References

[I-D.bormann-t2trg-stp] Bormann, C. and K. Hartke, "The Series Transfer Pattern (STP)", draft-bormann-t2trg-stp-03 (work in progress), April 2020.

[RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <https://www.rfc-editor.org/info/rfc7009>.

Appendix A. Usage of the Series Transfer Pattern

This section discusses how the diff query of the TRL defined in Section 5.2 is a usage example of the Series Transfer Pattern defined in [I-D.bormann-t2trg-stp].

A diff query enables the transfer of a series of TRL updates, with the Authorization Server specifying U <= N_MAX diff entries as the U

most recent updates to the portion of the TRL pertaining to a registered device.

For each registered device, the Authorization Server maintains an update collection of maximum N_MAX items. Each time the TRL changes, the Authorization Server performs the following operations for each registered device.

1. The Authorization Server considers the portion of the TRL pertaining to that registered device. If the TRL portion is not affected by this TRL update, the Authorization Server stops the processing for that registered device.

2. Otherwise, the Authorization Server creates two sets 'trl_patch' of token hashes, i.e. one "removed" set and one "added" set, as related to this TRL update.

3. The Authorization Server fills the two sets with the token hashes of the removed and added Access Tokens, respectively, from/to the TRL portion from step 1.

4. The Authorization Server creates a new series item including the two sets from step 3, and adds the series item to the update collection associated to the registered device.

When responding to a diff query request from a registered device (see Section 5.2), 'diff' is a subset of the collection associated to the requester, where each 'diff_entry' record is a series item from that collection. Note that 'diff' specifies the whole current collection when the value of U is equal to SIZE, i.e. the current number of series items in the collection.

The value N of the 'diff' query parameter in the diff query request allows the requester and the Authorization Server to trade the amount of provided information with the latency of the information transfer.

Since the collection associated to each registered device includes up to N_MAX series item, the Authorization Server deletes the oldest series item when a new one is generated and added to the end of the collection, due to a new TRL update pertaining to that registered device. This addresses the question "When can the server decide to no longer retain older items?" in Section 3.2 of [I-D.bormann-t2trg-stp].

Appendix B. Usage of the "Cursor" Pattern

Building on Appendix A, this section describes how the diff query of the TRL defined in Section 5.2 can be further improved by using the "Cursor" pattern of the Series Transfer Pattern (see Section 3.3 of [I-D.bormann-t2trg-stp]).

This has two benefits. First, the Authorization Server can avoid excessively big latencies when several diff entries have to be transferred, by delivering one adjacent subset at the time, in different diff query responses. Second, a requester can retrieve diff entries associated to TRL updates that, even if not the most recent ones, occurred after a TRL update indicated as checkpoint.

To this end, each series item in an update collection is also associated with an unsigned integer 'index', with value the absolute counter of series items added to that collection minus 1. That is, the first series item added to a collection has 'index' with value 0. Then, the values of 'index' are used as cursor information.

Furthermore, the Authorization Server defines an unsigned integer MAX_DIFF_BATCH <= N_MAX, specifying the maximum number of diff entries to be included in a single diff query response. If supporting diff queries, the Authorization Server SHOULD provide registered devices and administrators with the value of MAX_DIFF_BATCH, upon their registration (see Section 6).

Finally, the full query and diff query exchanges defined in Section 5.1 and Section 5.2 are extended as follows.

In particular, successul responses from the TRL endpoint MUST use the Content-Format "application/ace-trl+cbor" defined in Section 10.2 of this specification.

B.1. Full Query Request

No changes apply to what defined in Section 5.1.

B.2. Full Query Response

When sending a 2.05 (Content) response to a full query request (see Appendix B.1), the response payload includes a CBOR map with the following fields, whose CBOR labels are defined in Appendix B.4.4.

- 'trl': this field MUST include a CBOR array of token hashes. The CBOR array is populated and formatted as defined in Section 5.1.

- 'cursor': this field MUST include either the CBOR simple value Null or a CBOR unsigned integer.

  The CBOR simple value Null MUST be used to indicate that there are currently no TRL updates pertinent to the requester, i.e. the update collection for that requester is empty. This is the case from when the requester registers at the Authorization Server until a first update pertaining that requester occurs to the TRL.

  Otherwise, the field MUST include a CBOR unsigned integer, encoding the 'index' value of the last series item in the collection, as corresponding to the most recent update pertaining to the requester occurred to the TRL.

B.3. Diff Query Request

In addition to the query parameter 'diff' (see Section 5.2), the requester can specify a query parameter 'cursor', with value an unsigned integer.

B.4. Diff Query Response

The Authorization Server composes a response to a diff query request (see Appendix B.3) as follows, depending on the parameters specified in the request and on the current status of the update collection for the requester.

B.4.1. Empty Collection

If the collection associated to the requester has no elements, the Authorization Server returns a 2.05 (Content) diff query response.

The response payload includes a CBOR map with the following fields, whose CBOR labels are defined in Appendix B.4.4.

- 'diff': this field MUST include an empty CBOR array.

- 'cursor': this field MUST include the CBOR simple value Null.

- 'more': this fields MUST include the CBOR simple value False.

B.4.2. Cursor Not Specified in the Diff Query Request

If the update collection associated to the requester is not empty and the diff query request does not include the query parameter 'cursor', the Authorization Server returns a 2.05 (Content) diff query response.

The response payload includes a CBOR map with the following fields, whose CBOR labels are defined in Appendix B.4.4.

- 'diff': this field MUST include a CBOR array, containing L = min(U, MAX_DIFF_BATCH) diff entries. In particular, the CBOR array is populated as follows.

  * If U <= MAX_DIFF_BATCH, these diff entries are the last series items in the collection associated to the requester, corresponding to the L most recent TRL updates pertaining to the requester.

  * If U > MAX_DIFF_BATCH, these diff entries are the eldest of the last L series items in the collection associated to the requester, as corresponding to the first L of the U most recent TRL updates pertaining to the requester.

  The 'diff' CBOR array as well as the individual diff entries have the same format specified in Figure 4 and used for the reponse payload defined in Section 5.2.

- 'cursor': this field MUST include a CBOR unsigned integer. This takes the 'index' value of the series element of the collection included as first diff entry in the 'diff' CBOR array. That is, it takes the 'index' value of the series item in the collection corresponding to the most recent update pertaining to the requester and returned in this diff query response.

  Note that 'cursor' takes the same 'index' value of the last series item in the collection when U <= MAX_DIFF_BATCH.

- 'more': this field MUST include the CBOR simple value False if U <= MAX_DIFF_BATCH, or the CBOR simple value True otherwise.

  If 'more' has value True, the requester can send a follow-up diff query request including the query parameter 'cursor', with the same value of the 'cursor' field included in this diff query response. This would result in the Authorization Server transfering the following subset of series items as diff entries, i.e. resuming from where interrupted in the previous transfer.

B.4.3. Cursor Specified in the Diff Query Request

If the update collection associated to the requester is not empty and the diff query request includes the query parameter 'cursor' with value P, the Authorization Server proceeds as follows.

- The Authorization Server MUST return a 4.00 (Bad Request) response in case the 'cursor' parameter specifies a value other than 0 or than a positive integer.

- If no series item X with 'index' having value P is found in the collection associated to the requester, then that item has been previously removed from the history of updates for that requester (see Appendix A). In this case, the Authorization Server returns a 2.05 (Content) diff query response.

  The response payload includes a CBOR map with the following fields, whose CBOR labels are defined in Appendix B.4.4.

  * 'diff': this field MUST include an empty CBOR array.

  * 'cursor': this field MUST include the CBOR simple value Null.

  * 'more': this field MUST include the CBOR simple value True.

  With the combination ('cursor', 'more') = (Null, True), the Authorization Server is signaling that the update collection is in fact not empty, but that some series items have been lost due to their removal, including the item with 'index' value P that the requester wished to use as checkpoint.

  When receiving this diff query response, the requester should send a new full query request to the Authorization Server, in order to fully retrieve the current pertaining portion of the TRL.

- If the series item X with 'index' having value P is found in the collection associated to the requester, the Authorization Server returns a 2.05 (Content) diff query response.

  The response payload includes a CBOR map with the following fields, whose CBOR labels are defined in Appendix B.4.4.

  * 'diff': this field MUST include a CBOR array, containing L = min(SUB_U, MAX_DIFF_BATCH) diff entries, where SUB_U = min(NUM, SUB_SIZE), and SUB_SIZE is the number of series items in the collection following the series item X.

    That is, these are the L updates pertaining to the requester that immediately follow the series item X indicated as checkpoint. In particular, the CBOR array is populated as follows.

    + If SUB_U <= MAX_DIFF_BATCH, these diff entries are the last series items in the collection associated to the requester,

corresponding to the L most recent TRL updates pertaining to the requester.

+ If SUB_U > MAX_DIFF_BATCH, these diff entries are the eldest of the last L series items in the collection associated to the requester, corresponding to the first L of the SUB_U most recent TRL updates pertaining to the requester.

The 'diff' CBOR array as well as the individual diff entries have the same format specified in Figure 4 and used for the reponse payload defined in Section 5.2.

* 'cursor': this field MUST include a CBOR unsigned integer. In particular:

  + If L is equal to 0, i.e. the series item X is the last one in the collection, 'cursor' takes the same 'index' value of the last series item in the collection.

  + If L is different than 0, 'cursor' takes the 'index' value of the series element of the collection included as first diff entry in the 'diff' CBOR array. That is, it takes the 'index' value of the series item in the collection corresponding to the most recent update pertaining to the requester and returned in this diff query response.

  Note that 'cursor' takes the same 'index' value of the last series item in the collection when SUB_U <= MAX_DIFF_BATCH.

* 'more': this field MUST include the CBOR simple value False if SUB_U <= MAX_DIFF_BATCH, or the CBOR simple value True otherwise.

  If 'more' has value True, the requester can send a follow-up diff query request including the query parameter 'cursor', with the same value of the 'cursor' field specified in this diff query response. This would result in the Authorization Server transfering the following subset of series items as diff entries, i.e. resuming from where interrupted in the previous transfer.

B.4.4. TRL Parameters

This specification defines a number of fields used in the response to a diff query request to the TRL endpoint relying on the "Cursor" pattern, as defined in Appendix B.

The table below summarizes them, and specifies the CBOR key to use instead of the full descriptive name. Note that the Content-Format "application/ace-trl+cbor" defined in Section 10.2 of this specification MUST be used when these fields are transported.

Name	CBOR Key	CBOR Type	Reference
trl	TBD	array	[This Document]
cursor	TBD	simple value null / unsigned integer	[This Document]
diff	TBD	array	[This Document]
more	TBD	simple value True or False	[This Document]

Acknowledgments

The authors sincerely thank Carsten Bormann, Benjamin Kaduk, Jim Schaad, Goeran Selander and Travis Spencer for their comments and feedback.

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

Authors' Addresses

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

Email: marco.tiloca@ri.se

Ludwig Seitz
Combitech
Djaeknegatan 31
Malmoe SE-21135 Malmoe
Sweden

Email: ludwig.seitz@combitech.se

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

Email: francesca.palombini@ericsson.com

Sebastian Echeverria
CMU SEI
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
United States of America

Email: secheverria@sei.cmu.edu

Grace Lewis
CMU SEI
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
United States of America

Email: glewis@sei.cmu.edu