Workgroup: ACE Working Group
Internet-Draft: draft-ietf-ace-aif-06
Published: 5 March 2022
Intended Status: Standards Track
Expires: 6 September 2022
Authors: C. Bormann
Universität Bremen TZI

An Authorization Information Format (AIF) for ACE

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 6 September 2022.

Table of Contents

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).

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 traditionally is also 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 and the permissions 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

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) response by a Location-indicating mechanism to a request that the subject made to the resource listed (/a/make-coffee in the example shown in Table 2, which might return the location of a resource that allows GET to find out about the status 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 each number to the power of two 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

For application/aif+cbor:

Type name: application
Subtype name: aif+cbor
Required parameters: none
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: none
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

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

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

// 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. Some wider issues are discussed in [RFC8576].

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

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>.

[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>.

[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. 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