Stream: Internet Engineering Task Force (IETF)
RFC: 9237
Category: Standards Track
Published: August 2022
ISSN: 2070-1721
Author: C. Bormann
Universität Bremen TZI

RFC 9237
An Authorization Information Format (AIF) for Authentication and Authorization for Constrained Environments (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 Representational State Transfer (REST) resources identified by URI path.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9237.

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

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, ascertain that the authorization to request the operation does apply to the actual requester as authenticated, and 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 [RFC9200]) 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 the Constrained Application Protocol (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 Concise Data Definition Language (CDDL) [RFC8610] [RFC9165]. Terminology for constrained devices is defined in [RFC7228].

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

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. 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`) that 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 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 bit set, which in turn is 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, we use the URI of a resource on a CoAP server for the object identifier (`Toid`). More specifically, since the parts of the URI that identify the server ("authority" in [RFC3986]) are authenticated during REST resource access (Section 4.2.2 of [RFC9110] 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 REST-Specific 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 shown 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 it 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 [RFC9110] (the Location-indicating mechanisms). (The concept is somewhat comparable to "Access Control List (ACL) inheritance" in the Network File System version 4 (NFSv4) protocol [RFC8881], except that it does not use a containment relationship but rather 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 REST-Specific 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 Sections 2.1 and 2.3. As discussed, in this case the object identifier is specialized as a text string giving a relative URI (URI-local-part as the absolute path on the server serving as the 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 that has been 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
)
Dynamic-Offset = 32
```

Figure 4: AIF in CDDL

For the information shown in Table 1 and Figure 3, a representation in Concise Binary Object Representation (CBOR) [RFC8949] is given in Figure 5; again, several optimizations and 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 having chosen 32 as Dynamic-Offset means that all future CoAP methods that are registered can be represented both as themselves and in the Dynamic-X variant, but that 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

5.1. Media Types

IANA has added the following media types to the "Media Types" registry. The registration entries are in the following subsections.

Name	Template	Reference
aif+cbor	application/aif+cbor	RFC 9237, Section 4
aif+json	application/aif+json	RFC 9237, Section 4

Table 3: New Media Types

5.1.1. 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 "Sub-Parameter Registry for application/aif+cbor and application/aif+json" subregistry for `Toid`. Default value: "URI-local-part" (RFC 9237).

`Tperm`:
: the data type of a permission set for the object identified via a `Toid`. A value from the "Sub-Parameter Registry for application/aif+cbor and application/aif+json" subregistry for `Tperm`. Default value: "REST-method-set" (RFC 9237).

Encoding considerations: binary (CBOR)

Security considerations: Section 6 of RFC 9237

Interoperability considerations: N/A

Published specification: Section 4 of RFC 9237

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 9237, 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: N/A

Author/Change controller: IETF

Provisional registration: no

5.1.2. 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 subregistry for `Toid`. Default value: "URI-local-part" (RFC 9237).

`Tperm`:
: the data type of a permission set for the object identified via a `Toid`. A value from the media-type parameter subregistry for `Tperm`. Default value: "REST-method-set" (RFC 9237).

Encoding considerations: binary (JSON is UTF-8-encoded text)

Security considerations: Section 6 of RFC 9237

Interoperability considerations: N/A

Published specification: Section 4 of RFC 9237

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 9237, 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: N/A

Author/Change controller: IETF

Provisional registration: no

5.2. Registries

For the media types application/aif+cbor and application/aif+json, IANA has created a subregistry within [IANA.media-type-sub-parameters] for the media-type parameters `Toid` and `Tperm`, populated with the following:

Parameter	name	Description/Specification	Reference
Toid	URI-local-part	local-part of URI	RFC 9237
Tperm	REST-method-set	set of REST methods represented	RFC 9237

Table 4: New Media Type Parameters

The registration policy is Specification Required [RFC8126]. The designated expert will engage with the submitter to ascertain whether 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 they are in [KebabCase] so they can be easily translated into names used in APIs with popular programming languages.

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

5.3. Content-Format

IANA has registered Content-Format numbers in the "CoAP Content-Formats" subregistry, within the "Constrained RESTful Environments (CoRE) Parameters" registry [IANA.core-parameters], as follows:

Media Type	Encoding	ID	Reference
application/aif+cbor	-	290	RFC 9237
application/aif+json	-	291	RFC 9237

Table 5: New Content-Formats

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; Section 11.1 of [RFC7252] specifically applies if complex formats such as URIs are used for `Toid` or `Tperm`. 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
- 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

[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>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[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

[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] "Kebab Case", 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>.

[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>.

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, who 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