Abstract

Constrained Devices as they are used in the "Internet of Things" need security. 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, and need to ascertain that other devices they place requests on are the ones they intended.

To transfer detailed authorization information from an authorization manager (such as an ACE-OAuth Authorization Server) to a device, a compact representation format is needed. This document provides a suggestion for 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 refinement that describes REST resources (potentially dynamically created) and the permissions on 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 21 August 2021.
1. Introduction

Constrained Devices as they are used in the "Internet of Things" need security. 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, and need to ascertain that other devices they place
requests on 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 provides a suggestion for 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 refinement that describes REST resources (potentially dynamically created) and the permissions on them.

1.1. Terminology

This memo uses terms from [RFC7252] and [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]. 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. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

(Note that this document is itself informational, but it is discussing normative statements that MUST be put into concrete terms in each specification that makes use of this document.)

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 do not concern the AIF format with the subject for which the AIF data item is issued, so 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). 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.

\[
\text{AIF-Generic}\langle \text{Toid}, \text{Tperm} \rangle = [\ast [\text{Toid}, \text{Tperm}]]
\]

Figure 1: Definition of Generic AIF

In a specific data model, 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).

\[
\text{AIF-Specific} = \text{AIF-Generic}\langle \text{tstr}, \text{uint} \rangle
\]

Figure 2: Likely 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, the parts of the URI that identify the server ("authority" in [RFC3986]) are considered the realm of the authentication mechanism (which are handled in the cryptographic armor); we therefore focus on the "path-absolute" 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 unambiguous who is the target (enforcement point) of these
authorizations.

For the permissions ("Tperm"), we simplify the model permissions to giving the subset of the CoAP methods permitted. This model is summarized in Table 1.

<table>
<thead>
<tr>
<th>local-part</th>
<th>Permission Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>/s/temp</td>
<td>GET</td>
</tr>
<tr>
<td>/a/led</td>
<td>PUT, GET</td>
</tr>
<tr>
<td>/dtls</td>
<td>POST</td>
</tr>
</tbody>
</table>

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 and a LED actuator "/a/led" for read/write.

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], however, that requires some considerations of the ease and unambiguity of matching a given URI against a set of templates in an AIF object.

This simple information model also does not allow further conditionalizing 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. Extended REST-specific Model

The extended REST-specific model 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 [RFC7231] (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 the third limitation mentioned in Section 2.2.

+-----------------+----------------------------------+
| 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 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 extended model; 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 basic 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 (local-part as absolute path on the server serving as enforcement point). The permission set is specialized to a single number by the following steps:

* The entries in the table that specify the same 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 for Dynamic-DELETE).

* The set of numbers is converted into a single number 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.

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

Figure 3: An authorization instance encoded in JSON (46 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]:

```cddl
AIF-REST = AIF-Generic<path, permissions>
path = tstr ; URI relative to enforcement point
permissions = uint .bits methods
methods = &(
  GET: 0
  POST: 1
  PUT: 2
  DELETE: 3
)```

Bormann                  Expires 21 August 2021                 [Page 6]
A representation of this information in CBOR [RFC8949] is given in Figure 5; again, several optimizations/improvements are possible.

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].
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 "local-uri" for "Toid" and "REST-method-set" for "Tperm".

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 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

// RFC Ed.: please replace RFC XXXX with this RFC number and remove this note.

For "application/aif+cbor":

Type name: application
Subtype name: aif+cbor
Required parameters:
* "Toid": the identifier for the object for which permissions are supplied. A value from the subregistry for "Toid". Default value: "local-uri" (RFC XXXX).
* "Tperm": the data type of a permission set for the the object identified via a "Toid". Default value: "REST-method-set" (RFC XXXX).

Optional parameters: none
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: No known applications currently use this media type.
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:
  * "Toid": the identifier for the object for which permissions are supplied. A value from the subregistry for "Toid". Default value: "local-uri" (RFC XXXX).

  * "Tperm": the data type of a permission set for the object identified via a "Toid". Default value: "REST-method-set" (RFC XXXX).
Optional parameters: none
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: No known applications currently use this media type.
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

IANA is requested to create a registry for AIF with two sub-registries for "Toid" and "Tperm", populated with:

<table>
<thead>
<tr>
<th>Subregistry</th>
<th>name</th>
<th>Description/Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toid</td>
<td>local-part</td>
<td>local-part of URI as specified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in RFC XXXX</td>
</tr>
<tr>
<td>Tperm</td>
<td>REST-method-set</td>
<td>set of REST methods represented</td>
</tr>
<tr>
<td></td>
<td></td>
<td>as specified in RFC XXXX</td>
</tr>
</tbody>
</table>

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.

// RFC Ed.: please replace RFC XXXX with this RFC number and remove this note.

5.3. Content-Format

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

<table>
<thead>
<tr>
<th>Media Type</th>
<th>Content Coding</th>
<th>ID</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>application/aif+cbor</td>
<td>-</td>
<td>TBD1</td>
<td>RFC XXXX</td>
</tr>
<tr>
<td>application/aif+json</td>
<td>-</td>
<td>TBD2</td>
<td>RFC XXXX</td>
</tr>
</tbody>
</table>
6. Security Considerations

The security considerations of [RFC7252] apply. Some wider issues are discussed in [RFC8576].

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

* ensure that the cryptographic armor employed around this format fulfills the security objectives, and that the armor or some additional information included in it with the AIF information unambiguously identifies the subject to which the authorizations shall apply, and

* ensure that the types used for "Toid" and "Tperm" provide the appropriate granularity so that application requirements on the precision of the authorization information are fulfilled.

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

A generic 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 Section 2.3, it should either reject the AIF data item entirely or it should act only on the permissions that it does understand. In other words, the usual principle "everything is denied until it is explicitly allowed" should hold here as well.

7. References

7.1. Normative References


7.2. Informative References


Acknowledgements

Jim Schaad, Francesca Palombini, Olaf Bergmann, and Marco Tiloca provided comments that shaped the direction of this document. Alexey Melnikov pointed out that there were gaps in the media type specifications.

Author's Address

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