A brief introduction to xAAL v0.6-draft

Christophe Lohr Jérôme Kerdreux

June 16, 2017

Abstract

This document summaries information which specifies xAAL such as defined by the IHSEV/HAAL team of IMT-Atlantique (ex-Telecom Bretagne) in spring 2017.

Simply speaking, this release proposes to use CBOR in place of JSON for xAAL messages.

Be aware that this is an ongoing work.

1 Introduction

The xAAL system is a solution for home-automation interoperability. Simply speaking, it allows a device from vendor A (e.g., a switch) to talk to device from vendor B (e.g., a lamp).

The xAAL specification defines: (i) A functional distributed architecture; (ii) A mean to describe and to discover the interface and the expected behaviour of participating nodes by so-called schemas; and (iii) A secure communications layer via an IP (multicast) bus.

These points are detailed in the specification document of xAAL version 0.5-r2.[1] The reader should consider this document first.

The xAAL version 0.6 proposes to use the Concise Binary Object Representation (CBOR, rfc7049)[2] in place of JavaScript Object Notation (JSON, rfc7159)[3] for messages. The xAAL architecture and xAAL schemas remain the same as in version 0.5-r2.

2 Changes on the xAAL communication protocol

As in version 0.5-r2, the xAAL messages in version 0.6 are carried on an IP multicast bus (IPv4 or IPv6). Messages are still made of two layers: a *security layer* which encapsulates an *application layer*. Data, the associated semantic and behaviours are the same.

The novelty is that data are now serialized in CBOR. For instance, instead of JSON objects, one uses CBOR map with definite strings as key. Also, instead of using the textual representation of UUID (RFC 4122), one now uses its binary representation as a CBOR definite bytestring of 16 bytes.

2.1 The Security Layer

The data of the security layer is the payload of the UDP multicast messages of the xAAL bus. This is a CBOR map whose fields must be:

[1] `http://recherche.telecom-bretagne.eu/xaal/documentation/`

[2] `https://tools.ietf.org/html/rfc7049`

[3] `https://tools.ietf.org/html/rfc7159`

- Key: `"version"` – Value: the definite string `"0.6"`. (The version of the protocol.) Others values should be rejected.

- Key: `"targets"` – Value: a definite bytestring build as the CBOR serialization of the array of destination addresses for the message. Remember that xAAL addresses are UUID, encoded as definite bytestring[16]. A xAAL device receiving a message should accept it if its own xAAL address is present in the array of targets. An empty targets field means a broadcast message. A "targets" field containing an empty bytestring is not allowed (the message should be rejected).

- Key: `"timestamp"` – Value: an array of two (and exactly two) integers. The first number must be a CBOR unsigned int64; it is the number of seconds since the Epoch (1970-01-01 00:00:00 +0000 UTC). The second number must be a CBOR unsigned int32; it is the number of microseconds since the beginning of this second.

- Key: `"payload"` – Value: a definite bytestring which is the ciphered *application layer* according to version 0.5-r2 principles (Poly1305/Chacha20, a symmetric key, a binary nonce build on the timestamp of messages, an acceptance window for the timestamp of messages, the `targets` field covered by the cryptographic signature).

If a message includes other fields in addition to the above mandatory ones, the message may be accepted, and the extra fields must be ignored.

Keys and values of the above described CBOR map must have no CBOR tags.

2.2 The Application Layer

The application layer is a CBOR map whose fields must be: a `"header"` (see bellow), and an optionally `"body"`. Other fields must be ignored. Once again, keys and values of this map must not have CBOR tags.

The header. This is a CBOR ma whose fields must be:

- Key: `"source"` – Value: a definite bytestring[16]; the xAAL address (UUID) of the sender of the message.

- Key: `"devType"` – Value: a definite string; the schema name (the pair *classe.type*) of the sender.

- Key: `"msgType"` – Value: a definite string; the type of the message among `"request"` `"reply"` `"notify"`.

- Key: `"action"` – Value: a definite string; the name of the action brought by the message from the list of *methods* and *notifications* described in the considered schema. The considered schema is the one of the sender.

Other fields must be ignored.

The body. It contains parameters, values and information elements for queries, responses and notifications, if needed. The body is optional: this depends on what has been defined in the schema for the considered action.

If present, the body must be a CBOR map.

3 Changes on xAAL schemas

xAAL devices are described by so-called schemas. Schemas are document specifying attributes, notifications and methods.

Schemas are written in JSON for now. This is still the case with xAAL version 0.6. Version 0.6 proposes to replace JSON by CBOR for messages, not for schemas. There are several reasons for this:

- First, JSON cover the need. As indicated, a schema tells the list of attributes, notifications and methods of the device, plus some extra information to describe the device (vendor, model, etc.). The expressiveness of JSON is enough for this.

- Then, a schema should be a document readable by a human (and possibly editable), since this brings some semantics that should make sense to users and developers. A binary document, such as CBOR, is only understandable by software. This is not whishable in the context of xAAL schemas.

Unfortunately, mixing JSON and CBOR may leads to uncomfortable situations that are investigated bellow.

3.1 Specifying types of attributes and parameters

Schemas specify the types of the attributes and parameters of devices.

To do this, for each attribute or parameter description, a xAAL schema includes a fragment of Json-Schema. [4] [5]. A Json-Schema is a kind of grammar that specifies the form that a JSON data must fit. Note that compares to alternatives (e.g., Json Content Rules (JCR)[6], JSON Constrained Notation (JSCN)[7], et.), Json-Schema uses a pure JSON syntax. A Json-Schema specification is written in JSON. Thanks to this, the type of xAAL attributes and parameters (in Json-Schema) can be directly included in the xAAL schema.

The key point is that this describes a data presentation for data presented in JSON. But xAAL version 0.6 introduces messages in CBOR. Therefore, values of attributes and parameters are carried by xAAL messages, that is to say encoded in CBOR.

Two strategies may be considered to solve this situation where data are presented in CBOR despite a specification dedicated to JSON.

- In a future release of xAAL, Json-Schema may be replaced by the CBOR Data Definition Language (CDDL)[8]. Indeed, CDDL allows specifying the form that a CBOR data must fit. (Note that CDDL can specify CBOR as well as JSON data). Unfortunately, for now, there are few tools for CDDL (programming libraries in different language, software, etc.) This is why, in the meantime, xAAL version 0.6 follows the bellow strategy.

- xAAL version 0.6 defines its own transcoding rules between JSON and CBOR. As a consequence, xAAL schema may formalize data that should be in JSON strictly speaking, even if those data are in fact carried in CBOR. Ugly but convenient.

[4] http://json-schema.org/latest/json-schema-core.html
[5] http://json-schema.org/latest/json-schema-validation.html
[6] https://tools.ietf.org/html/draft-newton-json-content-rules-08
[7] https://tools.ietf.org/html/draft-miller-json-constrained-notation-00
[8] https://tools.ietf.org/html/draft-greevenbosch-appsawg-cbor-cddl-10

3.2 xAAL transcoding rules between CBOR and JSON

CBOR is a superset of JSON. CBOR introduces a high flexibility and expressiveness.

Here are some rules which restrict CBOR in the context of xAAL. The main idea (and the main need) is to be able to transcode CBOR data into JSON and vice-versa in a more-or-less equivalent way.

The basis of xAAL transcoding rules are those proposed by Section 4 of the rfc7049.[9] So, the reader is invited to consider this document first.

Some specificities:

- No CBOR tags. Tags should be ignored. One does not attempt to perform any specific transcoding choice on the basis of tags.
- Do its best to use the smallest CBOR size to encode numbers (integer and decimal), without loosing precision, if possible.
- No CBOR bignums nor bigfloats (Sections 2.4.2 2.4.3 of rfc7049). They are transcoded according to basic rules (i.e., as ordinary bytestring, as array of two integers...)
- Keys of CBOR map are definite strings. Fileds with other type of key are ignored.
- Definite CBOR bytestrings of a size of 16 bytes are (always) considered as UUID and are transcoded into JSON string using UUID textual representation (rfc4122).
- Others CBOR bytestrings are encoded in base64 in JSON strings.
- JSON strings that are a textual representation of an UUID are transcoded into a definite CBOR bytestring[16].
- Others JSON strings are transcoded into CBOR strings (without trying a base64 decoding since there is no rigorous certain way to know if a string is a base64 content or not).

Note those rules are not strictly symmetrical (see base64 issues). However, this is considered as fine enough. Authors of new xAAL schemas should keep those rules in mind and propose "simple" data.

Those transcoding are also convenient to integrate xAAL in web context (e.g. an HTML user interface), due to the high proximity of JSON and web technologies.

[9] `https://tools.ietf.org/html/rfc7049#section-4`