XEP-0335: JSON Containers

Matthew Wild
mailto:mwild1@gmail.com
xmpp:me@matthewwild.co.uk

2018-09-26
Version 0.1.1

Status	Type	Short Name
Deferred	Standards Track	NOT_YET_ASSIGNED

This specification defines an element to be used for encapsulating JSON data in XMPP.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

JSON is an increasingly popular format for data representation. While investigation has shown us that it cannot feasibly replace XML in all its uses (JSON Encodings for XMPP (XEP-0295) [1]) sometimes existing data is already in JSON or it is necessary to integrate with systems that use JSON while avoiding the overhead of marshalling that data to or from XML.
Traditional approaches have ranged from simply placing the XML into existing freeform text containers in XMPP (such as the message <body> tag or the presence <status> tag) to creating dedicated containers in a custom namespace. Neither of these approaches are ideal for preserving the extensibility or interoperability that XMPP provides.
This document aims to solve the problem by defining a standard way to embed JSON into any XMPP stanza, and even allowing its use with existing XMPP protocols where possible.

2 Requirements

This specification should:

- Allow stanza generators to unambiguously embed JSON within their stanzas.
- Allow stanza recipients a way to identify JSON content, and thus also validation.

3 Use Cases

Since JSON generally isn't designed for end-user presentation, most use-cases centre around JSON as part of machine-to-machine communication, or as part of a higher protocol, such as Publish-Subscribe (XEP-0060) [2].

Listing 1: JSON as a pubsub item payload

```
<message from='pubsub.shakespeare.lit' to='francisco@denmark.lit' id='
    foo'>
      <event xmlns='http://jabber.org/protocol/pubsub#event'>
        <items node='princely_musings'>
          <item id='ae890ac52d0df67ed7cfdf51b644e901'>
            <json xmlns="urn:xmpp:json:0">
            { "name": "romeo", "age": "421", "status": "single" }
            </json>
          </item>
        </items>
      </event>
    </message>
```

[1] XEP-0295: JSON Encodings for XMPP <https://xmpp.org/extensions/xep-0295.html>.
[2] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

Listing 2: JSON as part of a custom protocol

```
<iq to="recipient.example.com" type="get" id="q1">
      <query xmlns="http://example.com/user-queries">
        <json xmlns="urn:xmpp:json:0">
          { "name": "romeo" }
        </json>
      </query>
    </iq>
```

4 Business Rules

The <json> element MUST only contain character data, and the data MUST conform to RFC 4627 [3]. Specifically, the element MUST NOT be empty, as the empty string is not valid JSON. The data MUST be encoded as UTF-8 (though officially unspecified, this is the de facto encoding for JSON today).
Implementations SHOULD validate JSON they receive and intend to use, and be prepared to handle invalid data appropriately (such as by responding to the sender with the applicable XMPP error reply for the stanza type).
As the <json> element alone provides no context to the recipient about the kind of data it contains, only the format, it SHOULD always be encapsulated within another element that provides a context and SHOULD NOT be added as a direct child of a stanza.

5 Implementation Notes

The JSON container element is intended for communicating small pieces of generic JSON data within a particular context. XMPP entities MUST NOT attempt to interpret unexpected JSON data they receive, and servers SHOULD NOT inspect JSON data inside stanzas they are routing, other than for OPTIONAL validation.
When generating stanzas containing JSON payloads, implementations should be aware of the maximum stanza size considerations laid down in XMPP Core [4].
Embedding JSON is not intended as a substitute for the use of XML in XMPP, and no attempt should be made by protocol designers to use it as such. XMPP implementations are optimised for XML processing, and introducing mixed-format protocols on top of existing XMPP constructs could lead to performance and security considerations beyond the scope of this document.

[3] RFC 4627: The application/json Media Type for JavaScript Object Notation (JSON) <http://tools.ietf.org/html/rfc4627>.
[4] RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core <http://tools.ietf.org/html/rfc6120>.

6 Security Considerations

JSON is a very common format for data interchange today, and has many popular implementations available in browsers and standalone libraries that can be assumed relatively well-tested. However an implementation receiving JSON data from an untrusted entity should take precautions and MUST NOT attempt to use invalid JSON data it receives in any way, nor must it accept data in any encoding other than UTF-8 to avoid potential encoding mismatch attacks.

7 IANA Considerations

N/A.

8 XMPP Registrar Considerations

TBD.

9 XML Schema

This schema is descriptive, not normative.

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:json:0'
    xmlns='urn:xmpp:json:0'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-0335: http://www.xmpp.org/extensions/xep-0335.html
    </xs:documentation>
  </xs:annotation>

  <xs:element name='json' type='xs:string' minOccurs='0' maxOccurs='
      unbounded'/>
</xs:schema>
```