XEP-0432: Simple JSON Messaging

Dave Cridland
mailto:dave@hellopando.com
xmpp:dwd@dave.cridland.net

2022-04-12
Version 0.1.1

Status	Type	Short Name
Deferred	Standards Track	udt

This specification proposes a simple mechanism by which applications can transfer data safely, without needing additional protocol design work. It is intended to provide a protocol that is trivial to implement and can be driven with a simple API.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

Applications written on top of XMPP often need to exchange data that has no existing standard. Such applications are often written by developers unfamiliar with best practise in designing new extensions for XMPP, making it hard to achieve this simple design goal without causing longer term problems.
This leads to ”solutions” such as stuffing JSON directly in the <body/> element, for example, and recognising this at the receiver either by heuristics or by a special <subject/>. While this works, it is difficult to then migrate to something else, and enforces that custom clients are always used.
Therefore this document proposes a very simple (and simplistic) framework for sending such data which - while very light on features - nevertheless conforms to best practice, and yields an interoperable protocol. Unusually, this specification SHOULD NOT be used as a base upon which to build other standards.

1.1 Terminology

Data transferred using this specification is encoded using JSON. The type of the data is given by a URI under the same rules as an XML namespace, and this specification refers to this as the datatype.
Because this document defines mechanisms for sending essentially arbitrary data, no real-world examples are given.
Instead, example namespaces are used within an XML namespace prefixed by urn:example:

2 Overview

2.1 Discovering Support

Support for this protocol is advertised by the Service Discovery protocol defined in Service Discovery (XEP-0030) [1] using a feature of urn:xmpp:json-msg:0.
Support for a particular datatype is given by concatenating the urn:xmpp:json-msg:0 feature with a hash character (’#’) and the datatype, for example urn:xmpp:json-msg:0#urn:example:foo.

2.2 Data Transfers

Simple JSON Messaging payloads may also be placed within a <message/> stanza. <message/> stanzas MAY contain multiple UDT payloads, but typical usage is expected to be that there will be only one. The JSON Messaging payload may be ancillary data to another message, or a

[1] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

standalone message in its own right.

2.2.1 Protocol Syntax

A Simple JSON Messaging payload consists of a single element, <payload/>, qualified by the XML namespace urn:xmpp:json-msg:0. It has a single, mandatory attribute of datatype, which MUST contain a string conformant to the requirements for XML namespaces (typically a URI under the control of the application developer).
As with XML namespaces, this URI is never expected to be resolved, and is used solely as an identifier. Different strings are considered entirely different datatypes, and common prefixes etc MUST be considered irrelevant for the purposes of interpreting the data. There are no common or standard datatypes.
The <payload element contains exactly one mandatory child element, the <json/> element defined in JSON Containers (XEP-0335) [2]. This in turns contains the JSON data.

```
<message from="gamer@game-company.example"
    to="match-maker.game-company.example"
    id="12345">
  <payload xmlns="urn:xmpp:json-msg:0" datatype="urn:example:foo">
    <json xmlns="urn:xmpp:json:0">
      {
        "annoying-teenager-level": 11
      }
    </json>
  </payload>
</message>
```

3 API Requirements

In order to satisfy the goals of this protocol, client library developers are encouraged to provide a simple to use API for this protocol. Developers are encouraged to use terms such as "JSON Message" in their API calls and documentation.
Support for a particular datatype SHOULD be advertised automatically when listening for custom messages of that type if possible.

4 Schema

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

[2] XEP-0335: JSON Containers <https://xmpp.org/extensions/xep-0335.html>.

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="
    qualified" targetNamespace="urn:xmpp:json-msg:0" xmlns:xs="http://
    www.w3.org/2001/XMLSchema">
  <xs:element name="payload" type="udt:payloadType" xmlns:udt="
      urn:xmpp:json-msg:0"/>
  <xs:complexType name="payloadType">
    <xs:sequence>
      <xs:any minOccurs="1" maxOccurs="1"/>
      <!-- Always a XEP-0335 json element, but I can't figure that out
          . -->
    </xs:sequence>
    <xs:attribute type="xs:string" name="datatype"/>
  </xs:complexType>
</xs:schema>
```

5 Security Considerations

All security implications herein are those of the payload.

6 IANA Considerations

This XEP requires no interaction with the Internet Assigned Numbers Authority (IANA) [3].

7 XMPP Registrar Considerations

None.

8 Acknowledgements

The authors wish to share any credit with many members of the community, including Florian Schmaus, Daniel Gultsch, Georg Lukas, and others.

[3] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.