Network Working Group
Internet-Draft
Intended status: Experimental
Expires: July 31, 2007

R. Sayre
Mozilla Corporation

January 27, 2007

JSON Uniform Messaging Protocol (JUMP)
draft-sayre-jump-04.txt

Status of this Memo

Copyright Notice

Abstract

JUMP uses HTTP and a lightweight layout for JSON records to edit the Web.

Table of Contents

1. Introduction

JSON [7] provides an interoperable object serialization format capable of representing numbers, strings, arrays, and a wide range of Unicode characters. This specification defines a loosely-coupled protocol based on a small set of conventions for JSON records and a profile of HTTP.

2. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [8].

3. JUMP Records

An example JUMP record:

```
{
 "title": "Example",
  "text": "Text and <a href='http://example.com/'>link</a>.",
  "link": "http://example.com/my.html",
 "media": "http://example.com/my.jpg",
    "id": ["B1549145-55CB-4A6B-9526-70D370821BB5"],
  "type": ["object"],
  "sync": "88C3865F-05A6-4E5C-8867-0FAC9AE264FC",
  "tags": ["foo","bar"],
  "self": "http://example.com/42.jump",
  "edit": "http://example.com/edit.cgi/42"
}
```

Detailed field definitions can be found in Section 3.1.

No JUMP field is required for every record, but interoperability will increase as the number of standard fields increases. Thus, a record containing zero standard fields is very unlikely to interoperate with any given JUMP implementation. Records without 'title' or 'text'

fields are also unlikely to interoperate with an independently-developed JUMP implementation. Guidelines for extension fields are detailed in Section 3.2.

3.1. Standard Fields

3.1.1. The 'type' Field

The 'type' field denotes the type its containing object. JUMP records have a default type of "object".

```
"type": "object"
```

Annotated JUMP arrays (Section 4.1) MUST contain the value "array".

```
"type": "array"
```

The 'type' field can have an array as its value, in which case the containing object is considered to be the union of its values. For example, the following example shows a type field that contains the "array" value required by JUMP arrays:

```
"type": ["array", "quux-extendo"]
```

JUMP records and fields SHOULD contain general type values whenever possible, so that independent implementations can interoperate to some degree. Additionaly, type names should be suitable for use as a MIME parameter value.

3.1.2. Text Fields

Text fields are based on the Text Constructs found in RFC 4287, section 3.1 [9]. Text fields are to be interpreted as HTML.

'title' A text field containing the title of the record.

'text' A text field containing the content or description of the record.

3.1.3. Other Fields

@tbd. Not hard to predict. Borrow from email and Atom as needed.

'author' If 'author' has a simple value, its value is taken to be

the equivalent of the 'name' element specified in the Atom Syndication Format [9]. If its value is an object, the 'value' field corresponds to the author 'name'. 'email' and 'uri' are as specified in the Atom Syndication Format [9]. If the value is an array, there are multiple authors.

'contributor' As specified for author, with a different field name.

'edit' A URI [5] or IRI [6] used to edit the record. See Section 5 for a description of JUMP's HTTP-based editing protocol.

'id' One or more permanent identifiers associated with the record.

'lang' The natural language of the JSON record and its descendents. If the value of 'lang' is an object, it may itself contain a dictionary mapping keys to languages. The 'default' value covers any key not explicitly mentioned.

'link' A URI [5] or IRI [6] used to view the record in a Web browser.

'media' The URI [5] or IRI [6] of a media object associated with the record.

'published' A string (or object with string value) containing a timestamp formatted as an Atom Syndication Format date [9].

'self' A URI [5] or IRI [6] where a read-only version of the record is present.

'sync' A string (or object with string value) containing a token for clients to use when identifing versions of a record, as with the HTTP Etag header [10].

'updated' A string (or object with string value) containing a timestamp formatted as an Atom Syndication Format date [9]. Values corresponding to the record's HTTP Last-Modified date are acceptable here.

'tags' An array of strings or objects (with string values) indicating labels that have been assigned to the record.

3.2. Extension Fields

4. Arrays

Sequences of JUMP records are denoted using JSON array syntax.

```
[
{
  "title": "Example",
   "text": "Text and <a href='http://example.com/'>link</a>.",
   "link": "http://example.com/my.html",
  "media": "http://example.com/my.jpg",
     "id": ["B1549145-55CB-4A6B-9526-70D370821BB5"],
   "type": ["object"],
   "sync": "88C3865F-05A6-4E5C-8867-0FAC9AE264FC",
   "tags": ["foo","bar"],
   "self": "http://example.com/42.jump",
   "edit": "http://example.com/edit.cgi/42"
},
{
  "title": "Example 2",
   "text": "Text and <a href='http://example.com/'>link</a>.",
   "link": "http://example.com/my2.html",
  "media": "http://example.com/my2.jpg",
     "id": ["8B6373C7-DA75-4120-A9BE-30C4CDA3CB73"],
   "type": ["object"],
   "sync": "BC4DEFA3-BF50-428B-8606-B3230953642A",
   "tags": ["foo","bar"],
   "edit": "http://example.com/edit.cgi/43"
}
]
```

4.1. Annotated Arrays

JUMP arrays can be annotated by using an extra level of indirection.

```
{
 "type": ["array", "quux-extendo"],
"title": "Example Annotated Array",
 "text": "This is <a href='./foo'>html</a> by default.",
"quux-x": "some special extensions property",
"value": [
  {
    "title": "Example",
     "text": "Text and <a href='http://example.com/'>link</a>.",
     "link": "http://example.com/my.html",
    "media": "http://example.com/my.jpg",
       "id": ["B1549145-55CB-4A6B-9526-70D370821BB5"],
     "type": ["object"],
     "sync": "88C3865F-05A6-4E5C-8867-0FAC9AE264FC",
     "tags": ["foo","bar"],
     "edit": "http://example.com/edit.cgi/42"
  },
  {
    "title": "Example 2",
     "text": "Text and <a href='http://example.com/'>link</a>.",
     "link": "http://example.com/my2.html",
    "media": "http://example.com/my2.jpg",
       "id": ["8B6373C7-DA75-4120-A9BE-30C4CDA3CB73"],
     "type": ["object"],
     "sync": "BC4DEFA3-BF50-428B-8606-B3230953642A",
     "tags": ["foo","bar"],
     "edit": "http://example.com/edit.cgi/43"
  }
 ]
}
```

4.2. Nesting

JUMP records and arrays can be nested. In the following example, an annotated array contains 3 elements, the second of which is an array itself.

```
{
 "type": ["array", "quux-extendo"],
"title": "Example Annotated Array",
 "text": "This is <a href='./foo'>html</a> by default.",
"quux-x": "some special extensions property",
"value": [
  {
    "title": "Example",
     "text": "Text and <a href='http://example.com/'>link</a>.",
     "link": "http://example.com/my.html",
```

```
      "media": "http://example.com/my.jpg",
         "id": ["B1549145-55CB-4A6B-9526-70D370821BB5"],
```



```
       "type": ["object"],
       "sync": "88C3865F-05A6-4E5C-8867-0FAC9AE264FC",
       "tags": ["foo","bar"],
       "edit": "http://example.com/edit.cgi/42"
    },
    {
      "title": "Example 2",
       "text": "Text and <a href='http://example.com/'>link</a>.",
       "link": "http://example.com/my2.html",
      "media": "http://example.com/my2.jpg",
         "id": ["77FAFBB4-2BA4-4D8D-9920-CC6D610D71DA"],
       "type": ["array"],
       "sync": "66B38253-BF67-471F-85B8-3DA601986DB6",
       "tags": ["foo","bar"],
       "edit": "http://example.com/edit.cgi/43",
      "value": [
        {
          "title": "Example 2a",
           "text": "Example 2a text.",
             "id": "59F3938E-430E-4E72-88FC-432D4D248076",
           "link": "http://example.com/my2.html#sectionA",
          "media": "http://example.com/my2a.jpg"
        },
        {
          "title": "Example 2b",
           "text": "Example 2b text.",
             "id": "4065FDC9-05E8-42CD-A626-FC4CA27AF933",
           "link": "http://example.com/my2.html#sectionB"
        }
      ]
    },
    {
      "title": "Example 3",
       "text": "Text and <a href='http://example.com/'>link</a>.",
       "link": "http://example.com/my3.html",
      "media": "http://example.com/my3.jpg",
         "id": ["8B6373C7-DA75-4120-A9BE-30C4CDA3CB73"],
       "type": ["object"],
       "sync": "BC4DEFA3-BF50-428B-8606-B3230953642A",
```

```
      "tags": ["foo","bar"],
      "edit": "http://example.com/edit.cgi/43"
    }
  ]
  }
```


5. Editing

JUMP records are edited using HTTP methods, like all HTTP resources.

5.1. Editing Single Records

@tbd.

5.2. Editing Individual JSON Fields

It is sometimes desirable to edit a single JSON field, rather than replace the whole record. For example, given a record at the URI http://example.com/foo, it should be possible to edit or add a single field.

```
{
 "title": "Example 3",
  "text": "Text and <a href='http://example.com/'>link</a>.",
  "link": "http://example.com/my3.html"
}
```

To edit a field, the client uses the familiar slash syntax of URIs to denote the object hierarchy. (note: may change this, or make it configurable, but the general idea works...)

```
PUT /foo/title HTTP/1.1
            Host: example.com
            Content-Type: application/json
            Content-Length: 34

            "Example 3: This is the new title"
```

A successful request would result in the following JUMP record:

```
{
```

```
 "title": "Example 3: This is the new title",
  "text": "Text and <a href='http://example.com/'>link</a>.",
  "link": "http://example.com/my3.html"
}
```

The 'edit' field discussed in Section 3.1.3 provides a useful way compute URIs for the client, sparing them complex path calculations. (todo: example using 'edit' instead of picking out an array slice).

5.3. Editing Arrays

@tbd.

6. Common JUMP Record Types

6.1. jsCalendar

jsCalendar is a JSON formulation of hCalendar [1], which is based on iCalendar [4].

```
{
           "type": ["vevent"],
       "category": [],
          "class": "PUBLIC",
    "description": "A synonym for the 'text' field defined above.",
   "dtstart-text": "A Human readable description of the start date",
        "dtstart": "YYYY-MM-DDTHH:MM:SS+ZZ:ZZ",
    "dtend-text": "A Human readable description of the end date",
         "dtend": "YYYY-MM-DDTHH:MM:SS+ZZ:ZZ",
      "duration": "",
      "location": "In a van, down by the river.",
        "status": "TENTATIVE",
       "summary": "A synonym for the 'title' field defined above.",
           "uid": "04C0CFFF-10C6-4366-A08D-114D9391D22F",
           "url": "http://example.com",
  "last-modified": "YYYY-MM-DDTHH:MM:SS+ZZ:ZZ"
}
```

6.2. jsCard

jsCard is a JSON formulation of hCard [2], which is based on vCard [3]. The TITLE property from both hCard and vCard is "vtitle" in jsCard.

```
{
          "type": ["vcard"],
            "fn": "",
             "n": {
              "family-name": "",
               "given-name": "",
          "additional-name": "",
        "honorific-prefix": "",
        "honorific-suffix": ""
                  },
      "nickname": "",
   "sort-string": "",
           "url": "",
        "email": [{
                     "type": "",
                    "value": ""
                  }],
           "tel": [{
                     "type": "",
```



```
                    "value": ""
                  }],
           "adr": {
        "post-office-box": "",
      "extended-address": "",
        "street-address": "",
              "locality": "",
                "region": "",
           "postal-code": "",
          "country-name": ""
                },
        "label": "",
          "geo": {
               "latitude": "",
              "longitude": ""
                },
           "tz": "",
        "photo": "",
```

```
            "logo": "",
           "sound": "",
            "bday": "",
          "vtitle": "",
            "role": "",
             "org": {
         "organization-name": "",
         "organization-unit": [""],
                    },
        "category": [""],
            "note": "text",
           "class": "PUBLIC",
             "key": "... some base64 ...",
          "mailer": "a mailer",
             "uid": "04C0CFFF-10C6-4366-A08D-114D9391D22F",
        "rev-text": "A Human readable description of the revision date",
             "rev": "YYYY-MM-DDTHH:MM:SS+ZZ:ZZ"
 }
```

6.3. jsAtom

[tbd] A JSON formulation of hAtom.

7. Security Considerations

None.

8. IANA Considerations

JUMP records can be identified with one of two media types, 'application/jump' and 'application/jumparray'.

8.1. application/jump

MIME media type name: application

MIME subtype name: jump

Mandatory parameters: None.

Optional parameters:

"types": This parameter contains some or all of the type information from the internal JUMP records.

Encoding considerations: Identical to those of "application/json" as described in RFC4627 [7], Section 6.

Security considerations: As defined in this specification.

Interoperability considerations: n/a.

Published specification: This specification.

Applications that use this media type: No known applications currently use this media type.

Additional information:

Magic number(s): n/a

File extension: .jump

Macintosh File Type code: TEXT

Person and email address to contact for further information: Robert Sayre

Intended usage: COMMON

Author/Change controller: Robert Sayre

8.2. application/jumparray

MIME media type name: application

MIME subtype name: jumparray

Mandatory parameters: None.

Optional parameters:

"types": This parameter contains some or all of the type information from the internal JUMP records.

Encoding considerations: Identical to those of "application/json" as described in RFC4627 [7], Section 6.

Security considerations: As defined in this specification.

Interoperability considerations: n/a.

Published specification: This specification.

Applications that use this media type: No known applications currently use this media type.

Additional information:

Magic number(s): n/a

File extension: .jumpa

Macintosh File Type code: TEXT

Person and email address to contact for further information: Robert Sayre

Intended usage: COMMON

Author/Change controller: Robert Sayre

9. Normative References

[1] Celik, T. and B. Suda, "hCalendar", June 2005, <http://microformats.org/wiki/hcalendar>.

[2] Celik, T. and B. Suda, "hCard", June 2005, <http://microformats.org/wiki/hcard>.

[3] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, September 1998.

[4] Dawson, F. and Stenerson, D., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 2445, November 1998.

[5] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[6] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

[7] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[8] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[9] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, December 2005.

[10] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

Author's Address

Robert Sayre
Mozilla Corporation

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be

found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).