

Internet Engineering Task Force (IETF)
Internet-Draft
Intended Status: Standards Track
Expires: November 7, 2014

Phillip Hallam-Baker
Comodo Group Inc.
May 6, 2014

JSON Log Format (JSON-L)
draft-hallambaker-jsonl-01

Abstract

A log file format based on the JSON encoding is described.

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 <http://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."

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Definitions	3
1.1.	Requirements Language	3
2.	Requirements	3
2.1.	Append Only Updates	3
2.2.	Resynchronization	3
3.	Specification	4
3.1.	Resynchronization	4
4.	Tag Specification	4
5.	Security Considerations	5
6.	IANA Considerations	5
7.	Acnowledgements	6
8.	References	7
8.1.	Normative References	7
	Author's Address	7

1. Definitions

1.1. Requirements Language

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 [[RFC2119](#)].

2. Requirements

A log file consists of a sequence of entries. In a JSON-L log file, each entry is a JSON object [[RFC7159](#)].

2.1. Append Only Updates

One of the chief drawbacks to using JSON notation as a log file format is that a log file is by definition a sequence of entries and a JSON document MUST contain exactly one object. Further JSON objects and arrays both require start '[' and end ']' markers.

Use of the JSON notation as a log file format would thus require a process appending data to a log file to either omit the end markers completely or erase the existing end markers, append the new data the file and write new end markers.

Both approaches are highly unsatisfactory, the first violates the JSON encoding rules and the second introduces a risk of a race condition if two processes attempt to update the same log file at the same time.

An encoding that permits log entries to be appended to the end of a log file without modifying the existing contents is both safer and more efficient. Since writing data to an append only log file is a common requirement, most platforms already provide efficient and robust primitives to effect append-only updates.

2.2. Resynchronization

Resynchronization is a mechanism that allows a reader to detect the start of a log file entry.

Resynchronization permits a reader to recover in the case that a log entry is corrupted and/or support random access to log file entries.

Log files may grow to thousands or millions of entries. When dealing with log files of such size it is frequently desirable to skip forward or backwards to quickly locate an entry added at a specific time.

Since JSON objects MAY be of variable lengths and MAY include nested JSON objects, a log file viewer that supports such a random access feature requires a simple means of locating the start of the next entry.

3. Specification

A JSON-L log file consists of a sequence of zero or more JSON objects as specified in [[RFC7159](#)] separated by white space that includes at least one newline character.

JSON-L-text = *(object x-ws)

x-ws = *ws %x0A *ws

3.1. Resynchronization

Since control characters are not permitted inside JSON string values and JSON objects MUST be separated by commas inside a JSON array, the sequence '}' *ws LF *ws '{' can only occur at the end of one log entry and the start of the next:

Boundary = end-object x-ws begin-object

4. Tag Specification

JSON-L is limited to an encoding and does not impose semantics on the JSON tags or values.

The W3C Extended Log File format [[W3C.WD-logfile-960221](#)] defines a set of tags which MAY be used as tags in the JSON object encoding.

[W3C.WD-logfile-960221] specifies this example:

```
#Version: 1.0
#Date: 12-Jan-1996 00:00:00
#Fields: time cs-method cs-uri
00:34:23 GET /foo/bar.html
12:21:16 GET /foo/bar.html
12:45:52 GET /foo/bar.html
12:57:34 GET /foo/bar.html
```

The corresponding JSON-L encoding is:

```
{ "Version": 1.0,
  "Date": "12-Jan-1996 00:00:00"}

{"time": "00:34:23", "cs-method" : "GET", "cs-uri": "/foo/bar.html"}
{"time": "12:21:16", "cs-method" : "GET", "cs-uri": "/foo/bar.html"}
{"time": "12:45:52", "cs-method" : "GET", "cs-uri": "/foo/bar.html"}
{"time": "12:57:34", "cs-method" : "GET", "cs-uri": "/foo/bar.html"}
```

Although the JSON-L encoding is rather less compact than the WD-Logfile format, it is considerably more flexible. It is not necessary for every entry to specify every field being logged. A writer can add the information that is appropriate in the circumstances.

5. Security Considerations

The security considerations for JSON-L are essentially the same as those for the JSON encoding on which it is based.

Specifically, since JSON's syntax is borrowed from Javascript it is possible to use the "eval ()" function in that language to parse entries in a JSON-L log. This constitutes an unacceptable security risk as the text could contain executable code along with data declarations. The same risk may exist in other languages that provide similar functions that execute data as code.

6. IANA Considerations

The MIME media type for JSON-Log text is application/json-l.

Type name: application

Subtype name: json-l

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: binary

Security considerations: See [this], Section TBS.

Interoperability considerations: Described in [this]

Published specification: [this]

Applications that use this media type: None (so far)

Additional information:

 Magic number(s): n/a

 File extension(s): .jsonl

 Macintosh file type code(s): TEXT

Person & email address to contact for further information: Phillip
Hallam-Baker <phill@hallambaker.com>

Intended usage: COMMON

Restrictions on usage: none

Author: Phillip Hallam-Baker

Change controller: Phillip Hallam-Baker <phill@hallambaker.com>

Note: No "charset" parameter is defined for this registration. Adding
one really has no effect on compliant recipients.

7. Acknowledgements

This is a minor modification of the JSON encoding developed by
Douglas Crockford. Nico Williams provided useful advice on the
resynchronization scheme.

8. References

8.1. Normative References

- [W3C.WD-logfile-960221] Hallam-Baker, P, Behlendorf, B, "Extended Log File Format", W3C NOTE WD-logfile-960221, 21 February 1996.
- [RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", [RFC 7159](#), March 2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

philliph@comodo.com

