json
Internet-Draft
Intended status: Standards Track
Expires: March 21, 2015

N. Williams
Cryptonector

September 17, 2014

JavaScript Object Notation (JSON) Text Sequences
draft-ietf-json-text-sequence-07

Abstract

This document describes the JSON text sequence format and associated media type, "application/json-seq".

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."

This Internet-Draft will expire on March 21, 2015.

Copyright Notice

Table of Contents

1. Introduction and Motivation

The JavaScript Object Notation (JSON) [RFC7159] is a very handy serialization format. However, when serializing a large sequence of values as an array, or a possibly indeterminate-length or never-ending sequence of values, JSON becomes difficult to work with.

Consider a sequence of one million values, each possibly 1 kilobyte when encoded -- roughly one gigabyte. It is often desirable to process such a dataset in an incremental manner: without having to first read all of it before beginning to produce results. Traditionally the way to do this with JSON is to use a "streaming" parser, but these are neither widely available, widely used, nor easy to use.

This document describes the concept and format of "JSON text sequences", which are specifically not JSON texts themselves but are composed of (possible) JSON texts. JSON text sequences can be parsed (and produced) incrementally without having to have a streaming parser (nor streaming encoder).

1.1. Conventions used in this document

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. JSON Text Sequence Format

Two ABNF rules are used in the definition of JSON text sequences: one for parsers, and one for encoders. Two rules are provided to permit recovery by parsers from sequences where some the elements are truncated for whatever reason. The rule for parsers is specified in terms of octet strings which are then interpreted as JSON-texts if possible. The rule for encoders, on the other hand, assumes that sequence elements are not truncated.

2.1. JSON text sequence parsing

The ABNF [RFC5234] for the JSON text sequence parser is as given in Figure 1.

```
JSON-sequence = *(1*RS possible-JSON)
RS = %x1E; "record separator" (RS), see ISO 646-1991
possible-JSON = 1*(not-RS); attempt to parse as UTF-8-encoded
                          ; JSON-text (see RFC7159)
not-RS = %x00-1d / %x1f-ff; any octets other than RS
```

Figure 1: JSON text sequence ABNF

In prose: a series of octet strings, each containing any octet other than a record separator (RS) (0x1E) [ISO.646.1991], all octet strings separated from each other by RS octets. Each octet string in the sequence is to be parsed as a JSON-text.

If parsing of such an octet string as a JSON-text fails, the parser should nonetheless continue parsing the remainder of the sequence; the parser SHOULD report such failures so that applications may terminate processing if desired. Multiple consecutive RS octets do not denote empty sequence elements between them. Parsers MAY report about empty sequence elements.

2.2. JSON text sequence encoding

The ABNF for the JSON text sequence encoder is given in Figure 2.

```
JSON-sequence = *(RS JSON-text LF)
RS = %x1E; see ISO 646-1991
LF = %x0A; "line feed" (LF), see ISO 646-1991
JSON-text = <given by RFC7159>
```

Figure 2: JSON text sequence ABNF

In prose: any number of JSON texts, each preceded and followed by one or more ASCII RS characters and each followed by a line feed (LF).

Since ASCII RS is a control character it may only appear in JSON strings in escaped form (see [RFC7159]), and since RS may not appear in JSON texts in any other form, RS unambiguously delimits the start of any element in the sequence. RS is sufficient to unambiguously delimit all top-level JSON value types other than numbers. Following each JSON-text in the sequence with an LF serves to disambiguate JSON-texts consisting of numbers at the top-level.

2.3. Top-level numeric values

Parsers MUST check that any JSON-texts that are a top-level number include JSON whitespace ("ws" ABNF rule from [RFC7159]) after the number, otherwise the JSON-text may have been truncated. Parsers MUST drop JSON-text sequence elements that may have been truncated (see previous sentence), but MAY report such texts (including, optionally, the parsed text and/or the original octet string).

2.4. Incomplete JSON texts are not be fatal

Per- Section 2.1, JSON text sequence parsers SHOULD NOT abort when RS terminates an incomplete JSON text. Such a situation may arise in contexts where append-writes to log files are truncated by the filesystem (e.g., due to a crash, or administrative process termination).

2.5. Interoperability note

There exist applications which use a format not unlike this one, but using LF instead of RS as the separator, some even using no separator between JSON texts. JSON text sequence parsers MAY parse such sequences, but JSON text sequence encoders MUST adhere to the rules in Section 2.2.

3. Security Considerations

All the security considerations of JSON [RFC7159] apply. This format provides no cryptographic integrity protection of any kind.

There is no end of sequence indicator. This means that "end of file", "end of transmission", and so on, can be indistinguishable from truncation and/or arbitrary additions. Applications where this matters should denote end of sequence by convention (e.g., Content-Length in the Hypertext Transfer Protocol (HTTP) [RFC7230]), and anyways they should use protocols that provide at least integrity protection of application data (e.g., Transport Layer Security (TLS) [RFC5246]).

4. IANA Considerations

The MIME media type for JSON text sequences is application/json-seq.

Type name: application

Subtype name: json-seq

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: binary

Security considerations: See <this document, once published>, Section 3.

Interoperability considerations: Described herein.

Published specification: <this document, once published>.

Applications that use this media type: <by publication time <https://stedolan.github.io/jq> is likely to support this format>.

5. Acknowledgements

Phillip Hallam-Baker proposed the use of JSON text sequences for logfiles and pointed out the need for resynchronization. James Manger contributed the ABNF for resynchronization. Stephen Dolan created <https://github.com/stedolan/jq>, which uses something like JSON text sequences (with LF as the separator between texts on output, and requiring only such whitespace as needed to disambiguate on input). Carsten Bormann suggested the use of ASCII RS, and Joe Hildebrand suggested the use of LF in addition to RS for disambiguating top-level number values. Paul Hoffman shephered the Internet-Draft. Many others contributed reviews and comments on the JSON Working Group mailing list.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014.

[ISO.646.1991] International Organization for Standardization, "Information technology - ISO 7-bit coded character set for information interchange", ISO Standard 646, 1991.

6.2. Informative References

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

Author's Address

Nicolas Williams
Cryptonector, LLC

Email: nico@cryptonector.com