

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 15, 2014

N. Williams
Cryptonector
March 14, 2014

JavaScript Object Notation (JSON) Text Sequences
draft-williams-json-text-sequence-00

Abstract

This document describes the JSON text sequence format and associated media type.

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 September 15, 2014.

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.	Introduction and Motivation	3
1.1.	Conventions used in this document	3
2.	JSON Text Sequence Format	4
3.	Security Considerations	5
4.	IANA Considerations	6
5.	Normative References	7
	Author's Address	8

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, which would be roughly one gigabyte. If processing such a dataset requires first parsing it entirely, then the result is very inefficient and the processing will be limited by virtual memory. "Online" parsers help, but they are neither widely available or widely used, nor are they easy to use.

Ideally such datasets could be parsed and processed one element at a time. Even if each element must be parsed in a not-online manner due to local choice of parser, the result will usually be sufficiently online: limited by the size of the biggest element in the sequence rather than by the size of the sequence.

This document describes the concept and format of "JSON text sequences", which are specifically not JSON texts themselves.

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

The ABNF [[RFC5234](#)] for the JSON text sequence format is as follows:

```
JSON-sequence = *(JSON-text 1*(text-separator))  
text-separator = %x20 / %x09 / %x0A / %x0D / %2C  
JSON-text = <given by RFC7159>
```

Figure 1: JSON text sequence ABNF

A JSON text sequence is a sequence of JSON texts, each followed by JSON whitespace (see the 'ws' rule in the JSON ABNF) separator.

Requirements:

- o Encoders MUST emit one or more JSON whitespace separator characters after each JSON text in a sequence. Two contiguous separators do not denote an empty JSON text between them.
- o Parsers MUST be able to parser a JSON text sequence consisting of JSON texts which do not contain newlines (except, of course, escaped within strings), and which are separated by either newlines, or carriage return and newline character pairs (U+000D U+000A).

Recommendations:

- o Parsers SHOULD NOT emit outputs which do not correspond to arrays, objects or strings until the separator is read. For example, an input of 'truefalse' is not a valid sequence of two JSON values, true and false! Neither is 'true0' a valid sequence of true and zero. Some parsers might in fact accept such sequences, which creates an ambiguity that is resolved by requiring (see above) that encoders never omit the separator.

Options:

- o Parsers MAY parse sequences where the separator is missing between any two consecutive texts provided that no ambiguity results (namely: when the first of the two texts is an array, an object, or a string).
- o Encoders MAY have an option for encoding JSON texts "compactly", without using newlines in the encoding. This maximizes interoperability with JSON text sequence parsers that utilize non-incremental, non-online JSON text parsers.

3. Security Considerations

All the security considerations of JSON [[RFC7159](#)] apply.

JSON text sequence parsers based on non-incremental, non-online JSON text parsers will not be able to efficiently parse JSON texts in which newlines appear; attempting to parse such sequences with non-incremental, non-online JSON text parsers creates a compute resource exhaustion vulnerability.

Parsers which accidentally parse invalid sequences like 'truefalse' (as the same as 'true\nfalse') create a mildly dangerous ambiguity. Encoders must never produce such sequences; parsers should not accept them.

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: JSON text sequences have been used in applications written with the jq programming language.

5. 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.

Author's Address

Nicolas Williams
Cryptonector, LLC

Email: nico@cryptonector.com