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

T. Bray, Ed.
Textuality Services

June 13, 2014

The I-JSON Message Format
draft-ietf-json-i-json-01

Abstract

I-JSON is a restricted profile of JSON designed to maximize interoperability and increase confidence that software can process it successfully with predictable results.

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 December 15, 2014.

Copyright Notice

Table of Contents

1. Introduction

RFC7159 describes the JSON data interchange format, which is widely used in Internet protocols. For historical reasons, that specification allows the use of language idioms and text encoding patterns which are likely to lead to interoperability problems and software breakage, particularly when a program receiving JSON data uses automated software to map it into native programming-language structures or database records. RFC 7149 describes practices which may be used to avoid these interoperability problems.

This document specifies I-JSON, short for "Internet JSON". The unit of definition is the "I-JSON message". I-JSON messages are also "JSON texts" as defined in RFC7159 but with certain extra constraints which enforce the good interoperability practices described in that specification.

1.1. Terminology

The terms "object", "member", "array", "number", "name", and "string" in this document are to be interpreted as described in RFC 7159 [RFC7159].

1.2. 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 RFC 2119 [RFC2119].

2. I-JSON Messages

An I-JSON message is a JSON text, as defined by RFC 7159.

2.1. Encoding and Characters

I-JSON messages MUST be encoded using UTF-8 [RFC3629].

Object member names, and string values in arrays and object members, MUST NOT include code points which identify Surrogates or Noncharacters.

This applies both to characters encoded directly in UTF-8 and to those which are escaped; thus, "\uDEAD" is always illegal because it is an unpaired surrogate, while "\uD800\uDEAD" would be legal.

2.2. Numbers

Software which implements IEEE 754-2008 binary64 (double precision) numbers [IEEE754] is generally available and widely used. Implementations which generate I-JSON messages MUST NOT assume that receiving implementations can process numeric values with greater magnitude or precision than provided by those numbers. I-JSON messages SHOULD NOT include numbers which express greater magnitude or precision than an IEEE 754 double precision number provides, for example 1E400 or 3.14159265358979323846264338327 9.

In particular, an I-JSON sender MUST NOT expect a receiver to treat an integer whose absolute value is greater than 9007199254740991 (i.e., that is outside the range [-(2**53)+1, (2**53)-1]) as an exact value.

For applications such as cryptography, where exact interchange of much larger numbers is required, it is RECOMMENDED to encode them in JSON string values. This requires that the receiving program understand the intended semantic of the value.

2.3. Object constraints

Objects in I-JSON messages MUST NOT have members with duplicate names.

Implementations which generate I-JSON messages MUST NOT assume that the order of object members in those messages is available to software which receives them.

3. Software Behavior

When software reads data which it expects to be an I-JSON message, but the data violates one of the MUST constraints in the previous section (for example, contains an object with a duplicate key, or a UTF-8 encoding error), that software MUST NOT trust nor act on the content of the message.

Designers of protocols which use I-JSON messages SHOULD provide a way, in this case, for the receiver of the erroneous data to signal the problem to the sender.

4. Protocol-design Recommendations

I-JSON is designed for use in Internet protocols. The following recommendations apply to the use of I-JSON in such protocols.

4.1. Top-level Constructs

An I-JSON message can be any JSON object. However, there are software implementations, coded to the older [RFC4627] specification, which only accept JSON objects or JSON arrays at the top level of JSON texts. For maximum interoperability with such implementations, it is RECOMMENDED that protocol designers avoid the use of JSON texts which are neither objects nor arrays.

4.2. Must-ignore Policy

It is frequently the case that changes to protocols are required after they have been put in production. Protocols which allow the introduction of new protocol elements in a way that does not disrupt the operation of existing software have proven advantageous in practice.

Such a policy is often referred to as "Must-Ignore" and is expressed with language such as "When receiving software encounters a protocol element which it does not recognize, it MUST NOT change its behavior as a consequence, and in particular must not fail." The converse policy, often referred to as "Must-Understand", does not tolerate the introduction of new protocol elements, and while this has proven necessary in certain protocol designs, in general it has been found to be overly restrictive and brittle.

A good way to support the use of Must-Ignore in I-JSON protocol designs is to require that top-level protocol elements must be JSON objects, and to specify that members whose names are unrecognized MUST NOT produce behavior changes.

4.3. Time and Date Handling

Protocols often contain data items which are designed to contain timestamps or time durations. It is RECOMMENDED that in all such data items be expressed in in ISO 8601 format, as specified in [RFC3339].

4.4. Binary Data

When it is required that an I-JSON protocol element contain arbitrary binary data, it is RECOMMENDED that this data be encoded in base64url RFC4648, section 5 [RFC4648].

5. Acknowledgements

I-JSON is entirely dependent on the design of JSON, largely due to Douglas Crockford. The specifics were strongly influenced by the contributors to the design of RFC 7159 on the IETF JSON Working Group.

6. Security Considerations

All the security considerations which apply to JSON (see RFC 7159) apply to I-JSON. There are no additional security considerations specific to I-JSON.

7. Normative References

[IEEE754] IEEE, "IEEE Standard for Floating-Point Arithmetic", 2008, <http://grouper.ieee.org/groups/754/>.

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

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

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

Author's Address

Tim Bray (editor)
Textuality Services

Email: tbray@textuality.com
URI: https://www.tbray.org/