Internet Engineering Task Force (IETF)
Request for Comments: 7493
Category: Standards Track
ISSN: 2070-1721

T. Bray, Editor
Textuality Services
March 2015

The I-JSON Message Format

Abstract

I-JSON (short for "Internet 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 is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741[1].

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7493[2].

Copyright Notice

[1] https://www.rfc-editor.org/rfc/rfc5741.html#section-2
[2] http://www.rfc-editor.org/info/rfc7493
[3] http://trustee.ietf.org/license-info

Table of Contents

1. Introduction

RFC 7159 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 that 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 7159 describes practices that 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 RFC 7159 but with certain extra constraints that 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 that identify Surrogates or Noncharacters as defined by [UNICODE].

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

2.2. Numbers

Software that implements IEEE 754-2008 binary64 (double precision) numbers [IEEE754] is generally available and widely used. Implementations that generate I-JSON messages cannot 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 that express greater magnitude or precision than an IEEE 754 double precision number provides, for example, 1E400 or 3.14159265358979323846264338327 9.

An I-JSON sender cannot 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 that require the exact interchange of numbers with greater magnitude or precision, it is RECOMMENDED to encode them in JSON string values. This requires that the receiving program understand the intended semantic of the value. An example would be 64-bit integers, even though modern hardware can deal with them, because of the limited scope of JavaScript numbers.

2.3. Object Constraints

Objects in I-JSON messages MUST NOT have members with duplicate names. In this context, "duplicate" means that the names, after processing any escaped characters, are identical sequences of Unicode characters.

The order of object members in an I-JSON message does not change the meaning of an I-JSON message. A receiving implementation MAY treat two I-JSON messages as equivalent if they differ only in the order of the object members.

3. Software Behavior

A major advantage of using I-JSON is that receivers can avoid ambiguous semantics in the JSON messages they receive. This allows receivers to reject or otherwise disregard messages that do not conform to the requirements in this document for I-JSON messages. Protocols that use I-JSON messages can be written so that receiving implementations are required to reject (or, as in the case of security protocols, not trust) messages that do not satisfy the constraints of I-JSON.

Designers of protocols that 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. Recommendations for Protocol Design

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 value. However, there are software implementations, coded to the older specification [RFC4627], which only accept JSON objects or JSON arrays at the top level of JSON texts. For maximum interoperability with such implementations, protocol designers SHOULD NOT use top-level JSON texts that 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 that allow the introduction of new protocol elements in a way that does not disrupt the operation of existing software have proven advantageous in practice.

This can be referred to as a "Must-Ignore" policy, meaning that when an implementation encounters a protocol element that it does not recognize, it should treat the rest of the protocol transaction as if the new element simply did not appear, and in particular, the implementation MUST NOT treat this as an error condition. The converse "Must-Understand" policy 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 be ignored.

4.3. Time and Date Handling

Protocols often contain data items that are designed to contain timestamps or time durations. It is RECOMMENDED that all such data items be expressed as string values in ISO 8601 format, as specified in [RFC3339], with the additional restrictions that uppercase rather than lowercase letters be used, that the timezone be included not defaulted, and that optional trailing seconds be included even when their value is "00". It is also RECOMMENDED that all data items containing time durations conform to the "duration" production in Appendix A of RFC 3339, with the same additional restrictions.

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 a string value in base64url; see Section 5 of [RFC4648].

5. Security Considerations

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

Since I-JSON forbids the use of certain JSON idioms that can lead to unpredictable behavior in receiving software, it may prove a more secure basis for Internet protocols and may be a good choice for protocol designers with special security needs.

6. Normative References

[IEEE754] IEEE, "IEEE Standard for Floating-Point Arithmetic", IEEE 754-2008, 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, <http://www.rfc-editor.org/info/rfc>.

[RFC3339] Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002, <http://www.rfc-editor.org/info/rfc>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003, <http://www.rfc-editor.org/info/rfc>.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006, <http://www.rfc-editor.org/info/rfc>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, <http://www.rfc-editor.org/info/rfc>.

[RFC7159] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, March 2014, <http://www.rfc-editor.org/info/rfc>.

[UNICODE] The Unicode Consortium, "The Unicode Standard", <http://www.unicode.org/versions/latest/>.

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 in the IETF JSON Working Group.

Author's Address

Tim Bray (editor)
Textuality Services
Email: tbray@textuality.com
URI: https://www.tbray.org/