JSON Serialization for the Swaptacular Messaging Protocol

Evgeni Pandurksi

2022-08-07

Overview

This document specifies how Swaptacular Messaging Protocol messages are serialized to JSON documents (`"applicatoin/json"`).

Note: 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.

Required Message Fields

For every specific type of message defined by the Swaptacular Messaging Protocol's specification, all of the defined message fields MUST be present in the serialized JSON document as properties. In addition, a `"type"` property MUST exist, specifying the type of the message.

For example, the serialization of an `AccountPurge` message would look like this:

```
{
  "type": "AccountPurge",
  "debtor_id": 1234,
  "creditor_id": 5678,
  "creation_date": "2022-08-19",
  "ts": "2022-08-20T16:59:59Z"
}
```

Message Filed Types

The specification of the Swaptacular Messaging Protocol uses several different field types, which MUST be serialized to JSON values as follows:

int32 To JSON number.

MUST be formatted as integer. MUST NOT contain a decimal point (`.`) or an exponent (`e` or `E`).

int64 To JSON number.

MUST be formatted as integer. MUST NOT contain a decimal point (`.`) or an exponent (`e` or `E`).

Note for implementators: Even thought ECMAScript 2021 supports BigInt's, the standard JSON parser and serializer does not allow to correctly process numbers outside the safe range from `-(2 ** 53 - 1)` to `2 ** 53 - 1`.

float To JSON number.

MUST be formatted as floating point number. MUST contain a decimal point (`.`), or an exponent (`e` or `E`), or both. The reason for this requirement is to allow generic JSON parsers to easily differentiate integers from floats.

Note for implementators: The standard ECMAScript 2021 JSON serializer does not satisfy this requirement.

string To JSON string.

Non-ASCII characters SHOULD NOT be escaped using the `\uXXXX` syntax.

date-time To JSON string.

The ISO 8601 timestamp format MUST be used.

date To JSON string.

The ISO 8601 date format MUST be used (`YYYY-MM-DD`).

bytes To JSON string.

Each byte MUST be represented by exactly two hexadecimal *uppercase* characters (Base16 encoding).

Default Encoding

When messages are serialized in JSON format, and received as a byte-stream, without an explicitly prescribed encoding, UTF-8 encoding MUST be presumed.