BOPT

Binary Object Protocol for Transport

Jade Krabbe, 2021-03-08

Table of Contents

Basics.....3

Metadata.....3

Content.....4

Content Negotiation.....4

Content Compression.....4

Framing.....5

Endianness.....5

Examples.....6

Responses:.....6

Request:.....6

Arbitrary Message:.....7

Basics

The Binary Object Protocol for Transport (BOPT) is a protocol for transporting any sort of data. It is based on BSON, a binary serialization format for JSON documents[1]. The entirety of every BOPT frame is binary-encoded, making it easier to decode for client machines.

BOPT is also minimalist, only including three metadata fields and a content field that can either be a BSON object or a string. This design allows extensibility, as extra metadata tags can be added after being negotiated between client and server.

Metadata

BOPT Metadata is the BSON fields outside the content field. This BOPT specification only lists 3 metadata values. They are as follows;

- Type: The MIME type of the content field. Required.
- Path: The path to the content being referred to by the BOPT frame. Optional.
- Checksum: A SHA-256 checksum for the content field. Should be negotiated whether this is required, but using checksum is recommended.

Metadata can be extended by just adding more values, but negotiation is recommended, to ensure client and server interoperability. Metadata extension should only be used if it would not make sense to have the value in the content field, for instance an Authentication value. Extension Metadata tags must be prefixed by “x-“.

[1] http://bsonspec.org/

Content

The BOPT content field can either be a BSON object, or a text string. If the content field is a BSON object, the Type metadata field should specify the negotiated data type instead of a MIME type.

Content Negotiation

BOPT content negotiation is done over BOPT as well, using the type field in requests as an array of MIME types or a single MIME type which the client would accept from the server. In an API-like context, the type metadata field may be used to specify the structure type instead of a MIME type.

Content Compression

The content field in a BOPT frame may be compressed. In this case, there should be an extra metadata value (compression-algorithm) specifying the compression algorithm. It should be noted that compression may make the message transmit across the wire faster, but it burdens the client with decompression.

Framing

BOPT framing is done simply. The first 14 bytes of a BOPT frame are as follows: 4 bytes containing the string “BOPT”, 1 byte containing the integer value of the major BOPT version, 1 byte containing the integer value of the minor BOPT version, and 8 bytes containing the 64-bit integer value of the length of the rest of the frame. The implementation of BOPT should interpret the last 8 bytes of this header to read the correct number of bytes, and then validate the frame’s BSON data before interpreting it.

Due to the 64-bit length value in the header of a BOPT frame, the maximum size of a BOPT frame is 18,446,744,073,709,551,615 + 14 bytes. BOPT implementations should check the length value to make sure it is not more than the system could reasonably handle.

Endianness

In keeping with the BSON standard, all BOPT bytes are little-endian, including the frame header.

Examples

The following are examples of BOPT frames, in JSON format for easy viewing.

Responses:

```
{
     “type”: ”text/plain”,
     “content”: “Hello World!”
}
{
     “type”: “text/html”,
     “path”: “bitflipped.org/”,
     “checksum”:
“357ff623669d0abb28544fe2ddde6f5a42db1296688319db8b9ac74ad545338
b”,
     “content”: “<html> </html>”
}
{
     “type”: “application/x-bson”,
     “content”: {
          “message”: “Hello there!”
     }
}
```

Request:

```
{
     “type”: “text/html”
     “path”: “bitflipped.org/”
}
```

Arbitrary Message:

```
{
     “type”: “text/plain”
     “checksum”:
“334d016f755cd6dc58c53a86e183882f8ec14f52fb05345887c8a5edd42c87b
7”
     “content”: “Hello!”
}
```