Binary Application Record Encoding (BARE)

Jiri Vlasak

2022-07-20

What is BARE?

- Messages encoding for data storage or transmission
- The original author: Drew DeVault
- The main contributor of the updates: Victorien Elvinger
- A kind of pre-editor to put it together: Jiri Vlasak (myself)

Why BARE? (The goals)

- Concise messages
- A well-defined message schema
- Broad compatibility with programming environments
- Simplicity of implementation

For what is BARE? (Use-cases)

- Self-describing authentication tokens for web services
- Opaque messages for transmitting arbitrary state between unrelated internet services
- A representation for packets in an internet protocol
- A structured data format for encrypted or signed application messages
- A structured data format for storing data in persistent storage

Feedback in the DISPATCH mailing list: CBOR

- CBOR is complex with self-descriptive messages
- BARE has simple schema and concise messages

Feature	JSON	CBOR	BARE
Contains schema in messages	Yes	Yes	No
?-based	text	binary	binary

Feedback: Need to flesh out the schema language

- I am not sure why/how
- More feedback needed

Feedback: Overhead measurements

- https://git.sr.ht/~qeef/draft-devault-bare/tree/master/item/compare-bare-to-cbor.py
- 1,000 messages inspired by the "Appendix B. Example Company" of the BARE I-D
- "someone" with "name", "email", "address" (list of four strings), and "orders" (list of 10 to 100 orders, where an order has i64 "orderId" and i32 "quantity")

  For 1,000 messages:

  - Raw length, i.e. 100 % [Bytes]: 747933
  - BARE length [%]: 100.94
  - CBOR length [%]: 200.43
  - JSON length [%]: 325.60
  - BARE gzipped [%]: 26.57
  - CBOR gzipped [%]: 29.28
  - JSON gzipped [%]: 32.29

Feedback: Why BARE is different/better/interesting?

- Comparison to well-known: Avro, Protobufs, Cap'n proto, Thrift, FlatBuffers
- However, none of the above is standardized

BARE has:

- concise messages, no embedded schema
- octet-aligned messages with little-endian representation
- encoding is bijective when possible
- designed with the simplicity in mind

Comparison: Support for numbers

Protocol	uint	u8	u16	u32	u64
Avro					
Protobufs	Y			Y	Y
Cap'n proto		Y	Y	Y	Y
Thrift					
FlatBufers		Y	Y	Y	Y
BARE	Y	Y	Y	Y	Y

Protocol	int	i8	i16	i32	i64	f32	f64
Avro				Y	Y	Y	Y
Protobufs	Y		Y	Y		Y	Y
Cap'n proto		Y	Y	Y	Y	Y	Y
Thrift		Y	Y	Y	Y		Y
FlatBufers		Y	Y	Y	Y	Y	Y
BARE	Y	Y	Y	Y	Y	Y	Y

Comparison: Support for other primitive types

Protocol	bool	str	data[]	data[length]	void	enum
Avro	Y	Y	Y	Y	Y	Y
Protobufs	Y	Y	Y		Y	Y
Cap'n proto	Y	Y	Y		Y	
Thrift	Y	Y	Y			
FlatBufers	Y	Y			Y	Y
BARE	Y	Y	Y	Y	Y	Y

Comparison: Support for aggregate types

Protocol	optional	list[]	list[length]	map	union	struct
Avro		Y		Y	Y	Y
Protobufs	Y	Y		Y	Y	Y
Cap'n proto		Y			Y	Y
Thrift		Y		Y		Y
FlatBufers	Y	Y	Y		Y	Y
BARE	Y	Y	Y	Y	Y	Y

Comparison: Summary

- I got lost in Protobufs' variable-length integers and the compatibility between them.
- Sometimes it was hard to find quickly if the type is supported – I am very sorry if I messed the tables!
- Other protocols/frameworks/libraries usually include more features, but are these features really important for encoding?
- BARE clearly distincts primitive and aggregate types.

Feedback: Schema syntax for future extensions points

From “4. Application Considerations”:

```
type Message union {MessageV1 | MessageV2 | MessageV3}

type MessageV1 ...

type MessageV2 ...

type MessageV3 ...
```

And the later deprecation of the messages:

```
type Message union {MessageV2 = 1 | MessageV3}
```

Feedback: Some pre defined thing for common data such as time

From “Appendix D. Design Decisions”:

```
There is no date/time type
   Use u64 for timestamp or str.
```

For example, ISO 8601 and parsing with the standard library can be used.

Thank you

Work progress within IETF?

- I am new, so not completely sure, but
  - *an existing WG* is good,
  - *form a new WG* makes no sense in my opinion,
  - *AD-sponsored* is probably the best,
  - *ISE* was the first try, but BARE tends to be a standard,
  - not sure the *other* options, though.

Questions and Discussion