Everything You Ever Wanted to Know about Protobuf 3

Dragoș Carp
12.9.2017

Data Serialization

“... is the process of translating data structures or object state into a format that can be stored [...] or transmitted [...]” [1]

[1] https://en.wikipedia.org/wiki/Serialization

Data Serialization

“... is the process of translating data structures or object state into a format that can be stored [...] or transmitted [...]” [1]

3 Must-Have Characteristics:

[1] https://en.wikipedia.org/wiki/Serialization

Data Serialization

“... is the process of translating data structures or object state into a format that can be stored [...] or transmitted [...]” [1]

3 Must-Have Characteristics:

- Portable
  - Platform: register size, endianness, memory layout
  - Across languages
  - String representation

[1] https://en.wikipedia.org/wiki/Serialization

Data Serialization

“... is the process of translating data structures or object state into a format that can be stored [...] or transmitted [...]” [1]

3 Must-Have Characteristics:

- Portable
    - Platform: register size, endianness, memory layout
    - Across languages
    - String representation
- Compact

[1] https://en.wikipedia.org/wiki/Serialization

Data Serialization

“... is the process of translating data structures or object state into a format that can be stored [...] or transmitted [...]” [1]

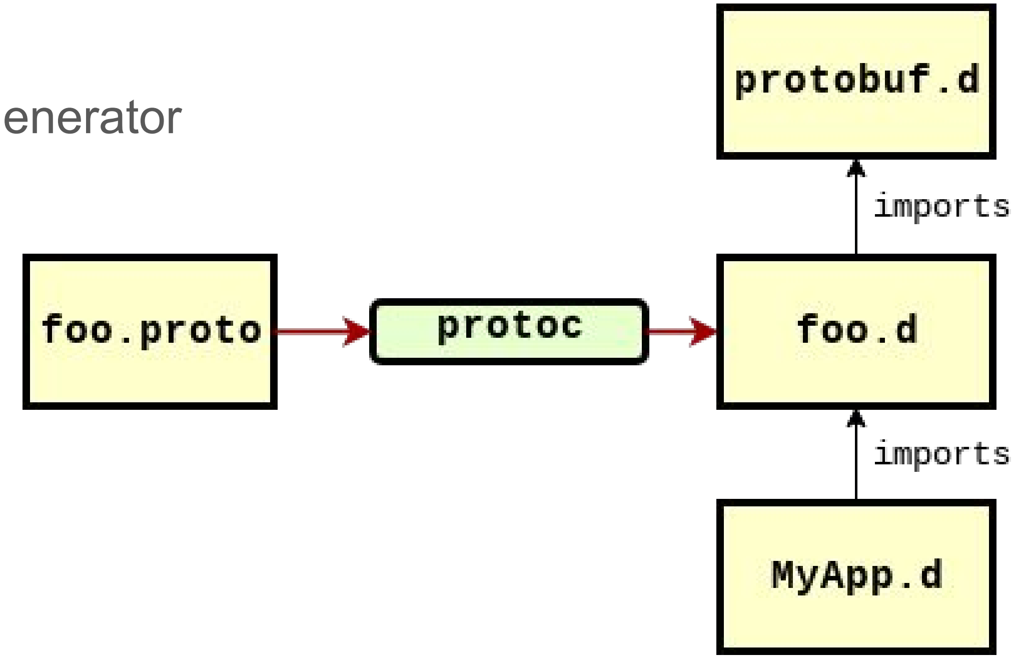
3 Must-Have Characteristics:

- Portable
  - Platform: register size, endianness, memory layout
  - Across languages
  - String representation
- Compact
- Extensible and Versionable

[1] https://en.wikipedia.org/wiki/Serialization

Protobuf Components

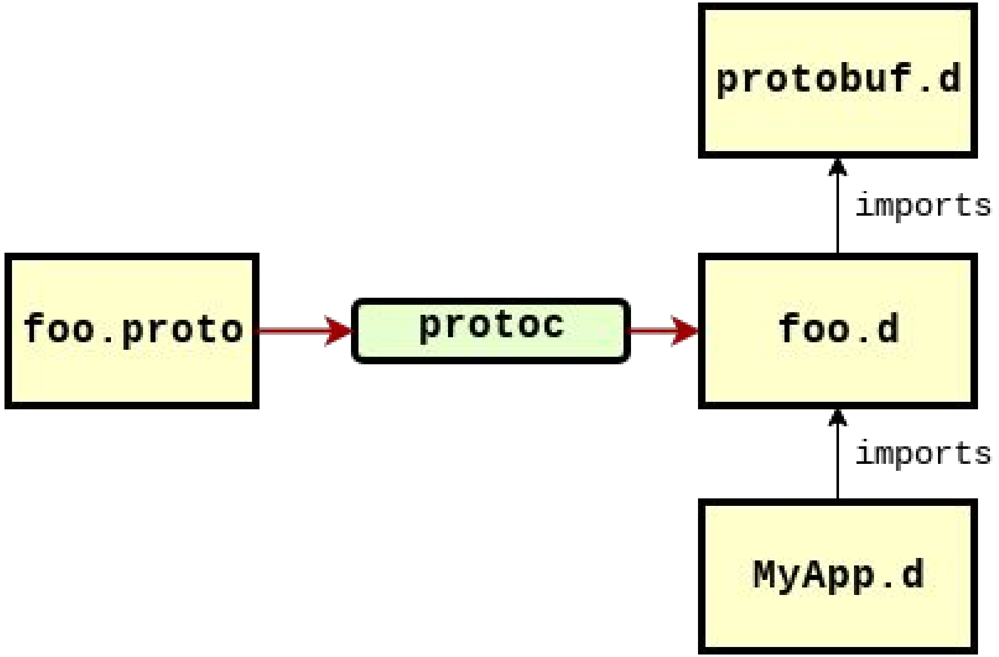
- Schema file
- Schema compiler
- Generated parser and generator
- Support library



Schema File

```
syntax = "proto3";

message SearchRequest {
  string query = 1;
  int32 page_number = 2;
  int32 result_per_page = 3;
}
```



```
% protoc --d_out=. requests.proto
```

Scalar Value Types

.proto	C++	Java	Python	D
double	double	double	float	double
float	float	float	float	float
int32	int32	int	int	int
int64	int64	long	int / long	long
uint32	uint32	int	int / long	uint
uint64	uint64	long	int / long	ulong
sint32	int32	int	int / long	int
sint64	int64	long	int / long	long

.proto	C++	Java	Python	D
fixed32	uint32	int	int / long	uint
fixed64	uint64	long	int / long	ulong
sfixed32	int32	int	int / long	int
sfixed64	int64	long	int / long	long
bool	bool	boolean	bool	bool
string	string	String	str / unicode	string
bytes	string	ByteString	str / bytes	ubyte[]

Default Values

Type	Default
double, float	`0.0`
int, long, etc.	`0`
bool	`false`
string	””
bytes	`[]`
enum	`0`
Message	`null / Message.init`

Enumerations

```
message SearchRequest {
  string query = 1;
  int32 page_number = 2;
  int32 result_per_page = 3;
  enum Corpus {
    UNIVERSAL = 0;
    WEB = 1;
    IMAGES = 2;
    LOCAL = 3;
    NEWS = 4;
    PRODUCTS = 5;
    VIDEO = 6;
  }
  Corpus corpus = 4;
}
```

OneOf

```
message SearchRequest {
  string query = 1;
  int32 page_number = 2;
  int32 result_per_page = 3;
  oneof corpus {
    Universal universal = 10;
    Web web = 11;
    Images images = 12;
    Local local = 13;
    News news = 14;
    Product product = 15;
    Video video = 16;
  }
}
```

Arrays

```
message Foo {
  int32 field1 = 1;
  repeated int32 field2 = 2;
  repeated int32 field3 = 3 [packed=true];
  repeated fixed32 field4 = 4 [packed=true];
}
```

Maps

```
message Foo {
  map<string, int> map1 = 1;
  map<int, string> map2 = 2;
  map<string, Foo> map3 = 3;
}
```

lowered to

```
message Foo {
  repeated Map1Entry map1 = 1;
  repeated Map2Entry map2 = 2;
  repeated Map3Entry map3 = 3;
}
message Map1Entry {
  string key = 1;
  int value = 2;
}
message Map2Entry {
  int key = 1;
  string value = 2;
}
message Map3Entry {
  string key = 1;
  Foo value = 2;
}
```

Encodings

Varint

`0`	`0000 0000`
`1`	`0000 0001`
`128`	`1000 0000  0000 0001`
`-1`	`1111 1111` … `0000 0001` (10 bytes)

Encodings

Varint

0	0000 0000
1	0000 0001
128	1000 0000 0000 0001
-1	1111 1111 … 0000 0001 (10 bytes)

Zigzag

0	0
-1	1
1	2
-2147483648	4294967295

Encodings

Varint

0	0000 0000
1	0000 0001
128	1000 0000 0000 0001
-1	1111 1111 … 0000 0001 (10 bytes)

Zigzag

0	0
-1	1
1	2
-2147483648	4294967295

Length-Delimited: <varint_length><data>

”foo”	0x03 0x66 0x6f 0x6f

Message Encoding

Sequence of `<field_key><field_value>`

`<field_key> = <field_tag><wire_type>` ← Varint encoded

`<field_value>` ← `<wire_type>` encoded

`wire_type`	Field Types
`0` - varint	int32, int64, uint32, uint64, sint32, sint64, bool, enum
1 - 64-bit	fixed64, sfixed64, double
2 - length-delimited	string, bytes, messages, packed repeated
5 - 32-bit	fixed32, sfixed32, float

Message Encoding Examples

```
message Foo1 {
    repeated int32 field1 = 1;
}
```

```
msg1.field1 = [100, 101];
  0x08 0x64 0x08 0x65
```

```
message Foo2 {
    string field1 = 1;
    string field2 = 2;
}
```

```
msg2.field2 = “foo”;
  0x12 0x03 0x66 0x6f 0x6f
```

```
message Foo3 {
    int32 field1 = 1;
    Foo2 field2 = 2;
}
```

```
msg3.field2.field2 = “foo”;
  0x12 0x05 0x12 0x03 0x66 0x6f 0x6f
```

D Implementation

protoc Generated Code: Enumerations

```
message SearchRequest {
  string query = 1;
  int32 page_number = 2;
  int32 result_per_page = 3;
  enum Corpus {
    UNIVERSAL = 0;
    WEB = 1;
    IMAGES = 2;
    LOCAL = 3;
    NEWS = 4;
    PRODUCTS = 5;
    VIDEO = 6;
  }
  Corpus corpus = 4;
}
```

```
class SearchRequest {
  @Proto(1) string query;
  @Proto(2) int page_number;
  @Proto(3) int result_per_page;
  enum Corpus {
    UNIVERSAL = 0,
    WEB = 1,
    IMAGES = 2,
    LOCAL = 3,
    NEWS = 4,
    PRODUCTS = 5,
    VIDEO = 6,
  }
  @Proto(4) Corpus corpus;
}
```

UDAs

```
struct Proto {
  uint tag;
  string wire;
  Flag!"packed" packed;
}

class Foo {
  @Proto(1) int field1;
};

__traits(getAttributes, Foo.field1);  // returns tuple(Proto(1))
```

`protoc` Generated Code: OneOf

```
message SearchRequest {
  string query = 1;
  oneof corpus {
    Web web = 11;
    News news = 14;
  }
}
```

```
class SearchRequest {
  @Proto(1) string query;
  enum CorpusCase {
    CorpusNotSet = 0,
    Web = 11,
    News = 14,
  };
  CorpusCase _corpusCase = CorpusCase.CorpusNotSet;
  @property CorpusCase corpusCase() { return _corpusCase; }
  void clearCorpus() { _corpusCase = CorpusCase.CorpusNotSet; }
  @Oneof("_corpusCase") union {
    @Proto(11) Web _web = defaultValue!(Web); mixin(oneofAccessors!_web);
    @Proto(14) News _news = defaultValue!(News); mixin(oneofAccessors!_news);
  }
}
```

`protoc` Generated Code: Arrays

```
message Foo {
  int32 field1 = 1;
  repeated int32 field2 = 2;
  repeated int32 field3 = 3 [packed=true];
  repeated fixed32 field4 = 4 [packed=true];
}
```

```
class Foo {
  @Proto(1) int field1;
  @Proto(2) int[] field2;
  @Proto(3, “”, Yes.packed) int[] field3;
  @Proto(4, “fixed”, Yes.packed) int[] field4;
}
```

protoc Generated Code: Maps

```
message Foo {
  map<string, int> map1 = 1;
  map<int, string> map2 = 2;
  map<string, Foo> map3 = 3;
}
```

```
class Foo {
  @Proto(1) int[string] map1;
  @Proto(2) string[int] map2;
  @Proto(3) Foo[string] map3;
}
```

`protoc` DLang support

Demo

google.protobuf API

```
auto toProtobuf(T)(T value)
  if (isAggregateType!T)

T fromProtobuf(T, R)(ref R inputRange, T result = defaultValue!T)
  if (isInputRange!R && isAggregateType!T)
```

Support Library Internals

```
template Message(T) {
  import std.meta : allSatisfy, staticMap, staticSort;
  import std.traits : getSymbolsByUDA;

  static assert(fields.length > 0, "Definition of '" ~ T.stringof ~
    "' has no Proto field");
  static assert(allSatisfy!(validateField, fields), "'" ~ T.stringof ~
    "' has invalid fields");

  alias fields = staticSort!(Less, unsortedFields);
  alias protos = staticMap!(protoByField, fields);

  alias fieldNames = staticMap!(fieldName, fields);

  private alias unsortedFields = getSymbolsByUDA!(T, Proto);
  private static enum fieldName(alias field) = __traits(identifier, field);
  private static enum Less(alias field1, alias field2) =
    protoByField!field1.tag < protoByField!field2.tag;
}
```

Support Library Internals

Demo

Well Known Types

- Any
- Timestamp
- Duration
- FieldMask
- Struct
- ListValue

Well Known Types

Demo

JSON Mapping

proto3	JSON Type
message	object
enum	string
map<key, value>	object
bool	false, true
string	string
bytes	Base64 string
int32, fixed32, uint32	number
int64, fixed64, uint64	string / number
float, double	number

proto3	JSON Type
Any	object `{“@type: “url”, “f1”: v1, …}`
Timestamp	string `“2017-09-12T19:52:32.021Z”`
Duration	string `“3.14159”`
Struct	object
Wrapper	JSON types
FieldMask	string
ListValue	array
Value	JSON value
NullValue	null

google.protobuf JSON API

```
JSONValue toJSONValue(T)(T value)
  if (isAggregateType!T)

T fromJSONValue(T)(JSONValue value, T result = defaultValue!T)
  if (isAggregateType!T)
```

Future steps

- Finish well-known types implementation
- Do some benchmarking
- Submit upstream
- Preserve unknown fields (protobuf 3.5)
- Add service definitions
- Preserve the comments

Thank you!