

Get these slides ⇒



Introduction to the protobuf ecosystem

FOR: IBANFIRST R&D PAYMENT TEAM

Théophile Roos

 [orness.com](https://www.orness.com)



ORNESS

Contents

1	What is protobuf ?	3
1.1	Syntax	4
1.1.1	Message definition	5
1.1.2	Enum definition	6
1.1.3	Enum definition (Aliases)	7
1.1.4	Message Composition	8
1.1.5	Nested Types	9
1.1.6	oneof	10
1.1.7	Packages	11
1.1.8	Service definition	12
1.1.9	Option and Annotation	13
1.2	Usefull links	14
2	RPCs	15
2.1	Protobuf based RPC	16
2.2	gRPC	17
2.3	gRPC Features/Concepts	18
2.4	gRPC-ecosystem	19
3	Tools	21
3.1	Official protobuf compiler: protoc	22
3.1.1	Issues with protoc	23
3.2	buf.build cli: buf	24
3.2.1	Migrating from protoc to buf	25
	with protoc	25
	with buf	25
	References	26

4	That's it!	27
---	------------	----

1 What is protobuf ?

Short for Protocol Buffers: an efficient, language-neutral data serialization format developed by Google

- Much smaller and faster than formats like JSON or XML
- Strong Typing and Schema: strict structure of the data defined in .proto files, enabling forward and backward compatibility
- Used for many use-cases: synchronous communication between network services, async communication through brokers systems (Kafka, RabbitMQ...), WebAssembly interfaces...

1.1 Syntax

Protobuf syntax versions

- proto2: legacy should not be used
- proto3: most used version currently
- revision <year>: new version with very limited support

First line of the .proto file

```
1 syntax = "proto3";  
2 // syntax = "proto2";  
3 // edition = "2023";
```

proto

no syntax specified: proto2 assumed

1.1.1 Message definition

Message definition

```
1  message User {
2      string name = 1 [json_name = "name"];
3      optional int32 age = 2 [default = 18];
4
5      // a comment
6      repeated string hobbies = 3;
7
8      map<string, int32> ibanToBalance = 4;
9  }
```

proto

Fields are defined with:

- a **FieldType**: `int32`, `string`, `bool`, `bytes`, etc.
- a **FieldName**: `name`, `age`, etc.
- a **Field Cardinality**: `optional`, `required`, `repeated`, etc.

notes:

- *maps can't be repeated*

1.1.2 Enum definition

```
1  enum USER_TYPE {
2      // a comment
3      USER_TYPE_UNSPECIFIED = 0;
4      USER_TYPE_REGULAR = 1;
5      USER_TYPE_GUEST = 2;
6      USER_TYPE_ADMIN = 3;
7  }
8
9  message User {
10     // ...
11     USER_TYPE user_type = 3;
12 }
```

The enum has a default value, which is the first value defined in the enum and with the FieldNumber 0.

1.1.3 Enum definition (Aliases)

```
1  enum Corpus {
2      option allow_alias = true;
3
4      CORPUS_UNSPECIFIED = 0;
5      CORPUS_UNIVERSAL = 1;
6      CORPUS_WEB = 2;
7      CORPUS_WWW = 2;
8  }
```

proto

Message and enums can have options defined with the option keyword.

1.1.4 Message Composition

```
1  // user.proto                                proto
2  import "user_type.proto";
3
4  message User {
5      // ...
6      USER_TYPE user_type = 3;
7  }
8
9  message Organisation {
10     string name = 1;
11     repeated User users = 2;
12 }
```

```
1  // user_type.proto                            proto
2  enum USER_TYPE {
3      // a comment
4      USER_TYPE_UNSPECIFIED = 0;
5      USER_TYPE_REGULAR = 1;
6      // ...
7  }
```

An *Organisation* contains a list of *User* which contain a field of type *USER_TYPE*

1.1.5 Nested Types

```
1  message User {
2      enum USER_TYPE {
3          USER_TYPE_UNSPECIFIED = 0;
4          USER_TYPE_REGULAR = 1;
5          // ...
6      }
7      // ...
8      USER_TYPE user_type = 3;
9  }
```

proto

Using nested type outside of the parent message definition

```
1  message UserProfile {
2      User.USER_TYPE type = 1;
3  }
```

proto

1.1.6 oneof

```
1  enum USER_TYPE {
2      // ...
3  }
4  enum ADMIN_TYPE {
5      // ...
6  }
7
8  message User {
9      // ...
10     oneof type {
11         USER_TYPE user_type = 3;
12         ADMIN_TYPE admin_type = 4;
13     }
14 }
```

proto

Using oneof to define a field that can be either a user or an admin.

Only one field of the oneof can be set at a time.

=> avoid using oneof if possible because of restrictions on the backward compatibility.

1.1.7 Packages

file structure:

```
1 ./foo/bar/baz.proto
2 ./fizz/buzz.proto
```

```
1 // foo/bar/baz.proto
2 package foo.bar;
3
4 message Baz {
5     // ...
6 }
```

proto

```
1 // fizz/buzz.proto
2 package fizz;
3
4 import "foo/bar/baz.proto";
5
6 message Buzz {
7     foo.bar.Baz baz = 1;
8 }
```

proto

1.1.8 Service definition

```
1  message GetUserRequest {
2      string username = 1;
3  }
4  message GetUserResponse {
5      User user = 1;
6  }
7
8  service UserService {
9      rpc GetUser(GetUserRequest) returns
10         (GetUserResponse);
11      // rpc DeleteUser(DeleteUserRequest)
12         returns (DeleteUserResponse);
13  }
```

A service *UserService* is defined with an RPC method *GetUser* that takes a *GetUserRequest* and returns a *GetUserResponse*.

1.1.9 Option and Annotation

Protobuf has a way to add metadata to the messages and fields.

```
1 message User {  
2     string name = 1 [json_name = "name"];  
3 }
```

proto

The `json_name` field option is used to define the name of the field when it is serialized to JSON.

```
1 import "google/protobuf/  
  descriptor.proto";  
2  
3 extend google.protobuf.FieldOptions {  
4     string field_suffix = 12345; // this  
    must be unique  
5 }  
6 message User {  
7     string name = 1 [(field_suffix)= "This  
    is the user's name"];  
8 }
```

proto

1.2 Usefull links

- Official documentation: <https://protobuf.dev/overview/>
- Un-official (but very good) documentation by the Buf team: <https://protobuf.com/docs/introduction>
- Third party protoc code generation plugins: https://github.com/protocolbuffers/protobuf/blob/main/docs/third_party.md
- Registered protobuf extension: <https://github.com/protocolbuffers/protobuf/blob/main/docs/options.md#existing-registered-extensions>
- somewhat standard google protobuf definitions: <https://github.com/googleapis/googleapis>

2 RPCs

Remote Procedure Calls: a way to call a function on a remote server

RPC based communication is pattern with many different implementations [1]:

- NFS: Network File System
- JSON-RPC: an RPC protocol that uses JSON-encoded messages.
- gRPC is a modern open source RPC framework that uses HTTP/2 and Protocol Buffers.
- IBM AIX: use of RPC in the AIX operating system. [2]

2.1 Protobuf based RPC

Some commonly used RPC framework

Framework	Transport Layer	Code Generation	Notable Features
gRPC	HTTP/2	Yes	Streaming, TLS, strong ecosystem
ConnectRPC	HTTP	Yes	Streaming, TLS, wide platform support
Ranger RPC	HTTP (any version)	Yes	Simple, fast, Go-centric
Apache Thrift	Any	Yes	Simple, fast, C++-centric
trpc	Any	No	Pure TypeScript

Side notes: gRPC can also use alternative serialization formats but it's uncommon.

2.2 gRPC

- gRPC: “gRPC Remote Procedure Calls”
- Started by Google in 2015, now a project of the [Cloud Native Computing Foundation](#)
- Transport layer: HTTP/2
- Interface Definition Language: Protocol Buffers
- Open-Source: Apache License 2.0
- Most used RPC framework
- Development by RFCs: <https://github.com/grpc/proposal>

Docs: <https://grpc.io>

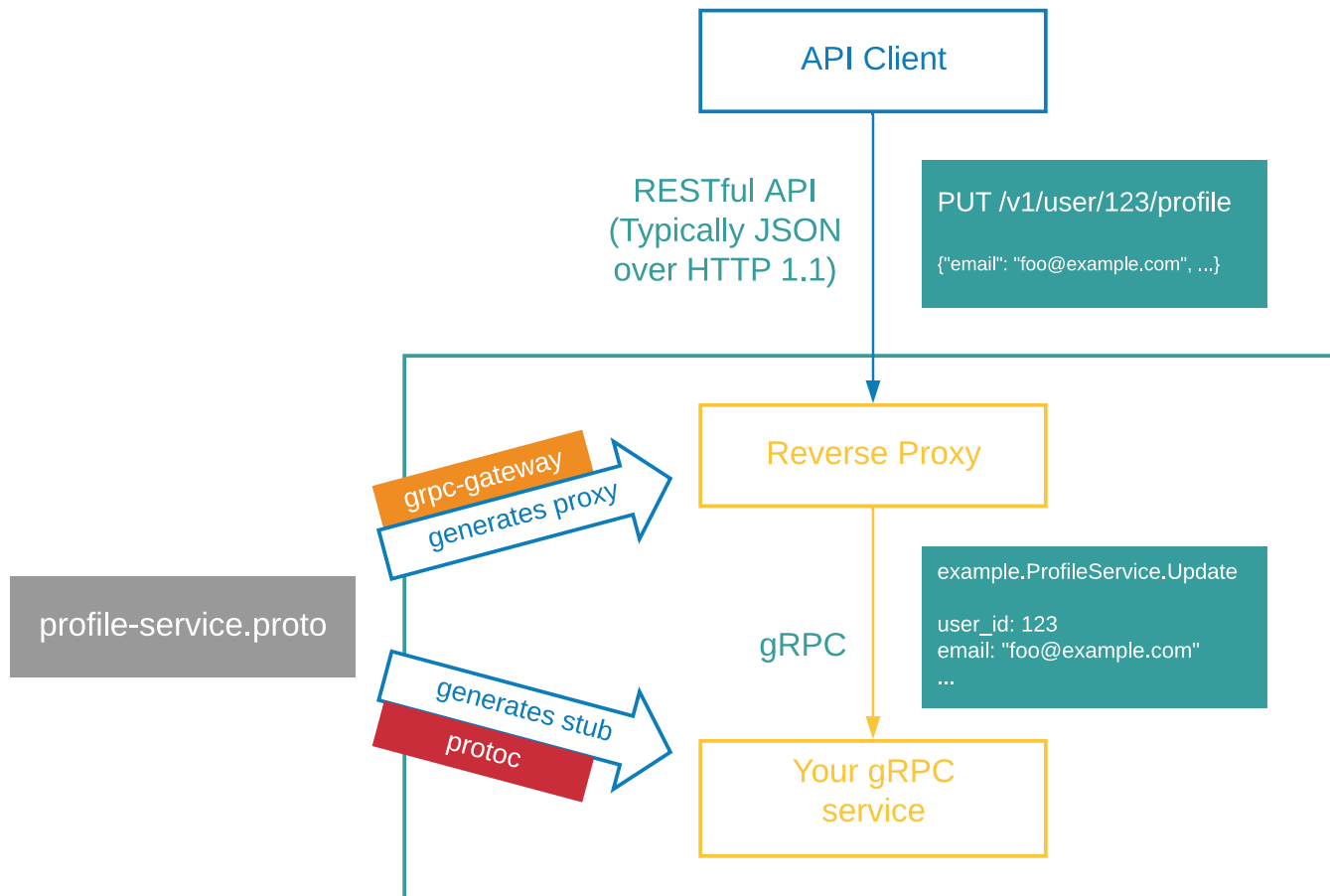
2.3 gRPC Features/Concepts

- Unary and Streaming RPC (client, server and bidirectional)
- Metadata (headers)
- Interceptors (middleware)
- Trailing metadata (HTTP/2 Trailers): used to send status codes, errors, etc.
- Deadlines/Timeouts: client communication timeouts to the server, which can know much time is left to respond.
- Cancellation: can be done by the client or the server, works on the server stops when the RPC is cancelled.
- Proxyless Load balancing: gRPC supports load balancing directly on the client side with the xDS protocol (xDS-based service discovery using a control plane)

2.4 gRPC-ecosystem

Suite of tools and libraries for gRPC

- *gRPC-gateway* : tool that converts gRPC APIs into RESTful APIs. Used to support legacy clients that are not capable of using gRPC.



- *openapiv2* : OpenAPI v2 specification generator for gRPC services.
- *grpc-opentracing* : OpenTracing instrumentation for gRPC.
- *grpc-health-probe* : gRPC health checking service.

Docs: <https://github.com/grpc-ecosystem/>

3 Tools

- **protoc**: official compiler, usually used with makefiles.
 - pros: official implementation, very flexible (almost to much).
 - cons: somewhat slow, makefiles are not fun, no linting, mandatory vendoring.
- **prototool** [3]: deprecated but still used protobuf toolchain created by Uber.
 - pros: fast, good integration with IDEs, linting.
 - cons: no package management, deprecated and unmaintained.
- **buf**: new protobuf toolchain, created by buf.build [4].
 - pros: fast, good integration with IDEs, linting, package management.
 - cons: remote plugins send your code to their cloud

3.1 Official protobuf compiler: protoc

Open-Source Code: <https://github.com/protocolbuffers/protobuf> . Use plugins to generate code and artifacts from .proto files.

```
1 $ ls
2 ./proto/foo/bar/baz.proto # our proto
3 ./third_party/googleapi/annotation.proto
4 # vendored protos
```

```
1 protoc --proto_path=./protos \
2 --proto_path=./third_party \
3 --go_out=paths=source_relative:. \
4 --go-grpc_out=paths=source_relative:. \
```

calls (args passed via stdin):

- protoc-gen-go
- protoc-gen-go-grpc

Proto files are managed manually (vendored proto code see [5])

3.1.1 Issues with protoc

Version/pinning issues:

- plugins versions are not pinned (you manage yourself your installation)
- dependencies are not pinned (you manage yourself the vendoring)
- homemade makefiles and/or shell scripts are required to automate the compilation
- (protoc version depends on your package manager or installation method)

Quality of code issues:

- No linting
- Modern IDE support is almost inexistent

3.2 buf.build cli: buf

All in one tool to manage your protobuf code using configuration first approach.

Version/pinning issues:

- ~~plugins versions are not pinned (you manage yourself your installation)~~
caveat: this require remote plugins
- ~~dependencies are not pinned (you manage yourself the vendoring)~~
- ~~homemade makefiles and/or shell scripts are required to automate the compilation~~
- ~~(protoc version depends on your package manager or installation method)~~

caveat: this require pinning the buf cli

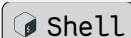
Quality of code issues:

- ~~No linting~~
- ~~Modern IDE support is almost inexistent~~ <https://buf.build/docs/cli/editor-integration/>
- + Breaking change detection, gRPC curl ...

3.2.1 Migrating from protoc to buf

with protoc

```
1 protoc --proto_path=./protos \  
2   --proto_path=./third_party \  
3   --go_out=paths=source_relative:. \  
4   --go-grpc_out=paths=source_relative:. \
```

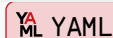


calls (args passed via stdin):

- protoc-gen-go
- protoc-gen-go-grpc

with buf

```
1 #buf.yaml  
2 version: v2  
3 modules:  
4   - path: proto  
5 deps:  
6   - buf.build/googleapis/googleapis  
7   - buf.build/bufbuild/protovalidate  
8 lint:  
9   use: [ "STANDARD"]  
10  
11 # buf.gen.yaml  
12 version: v2  
13 plugins:  
14   - name: go # managed plugin  
15     out: .  
16     opt: paths=source_relative  
17   - name: go-grpc # managed plugin  
18     out: .  
19     opt: paths=source_relative
```

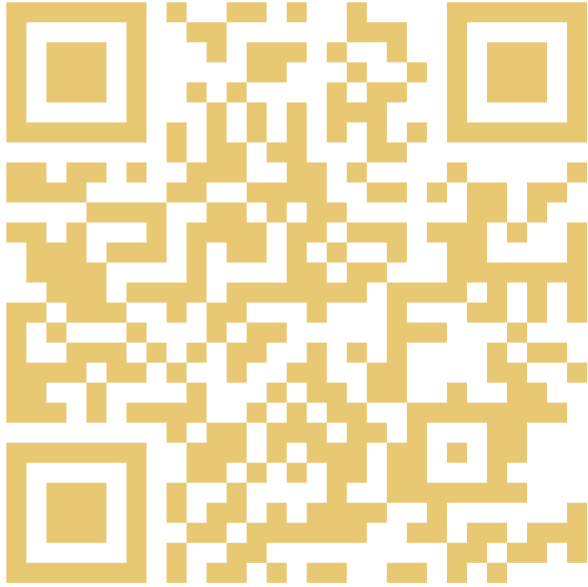


generate Go code with buf generate

References

- [1] Contributors to Wikimedia projects, “Remote procedure call - Wikipedia.” [Online]. Available: https://en.wikipedia.org/w/index.php?title=Remote_procedure_call&oldid=1292581498
- [2] “AIX.” [Online]. Available: <https://www.ibm.com/docs/en/aix/7.3.0?topic=call-rpc-features>
- [3] “prototool.” [Online]. Available: <https://github.com/uber/prototool>
- [4] “Buf - Crunchbase Company Profile.” [Online]. Available: <https://www.crunchbase.com/organization/buf-technologies#people>
- [5] “Compilation and Descriptors \vert The Protobuf Language”. [Online]. Available: <https://protobuf.com/docs/descriptors#standard-imports>
- [6] “Releases *c* · protocolbuffers/protobuf”. [Online]. Available: <https://github.com/protocolbuffers/protobuf/releases/tag/v31.1>
- [7] “protobuf.” [Online]. Available: <https://github.com/protocolbuffers/protobuf>
- [8] “Protocol Buffers.” [Online]. Available: <https://protobuf.dev/>
- [9] “Introduction \vert The Protobuf Language”. [Online]. Available: <https://protobuf.com/docs/introduction>
- [10] “Introduction - Buf Docs.” [Online]. Available: <https://buf.build/docs/cli>

4 That's it!



↑ Get these slides

Get in touch 🖐️

🌐 <https://orness.com>